DISTRIBUTED TASK ALLOCATION METHODOLOGIES FOR SOLVING THE INITIAL FORMATION PROBLEM

A Thesis
Presented to
The Academic Faculty

by

Luis Antidio Viguria Jimenez

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in Electrical and Computer Engineering in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
August 2008

DISTRIBUTED TASK ALLOCATION METHODOLOGIES FOR SOLVING THE INITIAL FORMATION PROBLEM

Approved by:

Professor Ayanna MacCalla Howard,
Advisor
School of Electrical and Computer
Engineering
Georgia Institute of Technology

Professor Bonnie Heck Ferri
School of Electrical and Computer
Engineering
Georgia Institute of Technology

Professor Tucker Balch
School of Interactive and Intelligent
Computing
Georgia Institute of Technology

Date Approved: 30 June 2008

To my family, specially my parents, and my friends who have support me over the last two years that I have been away from home.

ACKNOWLEDGEMENTS

First, I would like to thank my advisor, Ayanna M. Howard, for all her support and guidance these last two years. This thesis would not have come to the form and shape it is today without her. Also, I am very grateful to all the members of HumAnS Lab for their unselfish help and useful comments.

I am grateful to Dr. Magnus Egerstedt for his insightful discussions. Also, I would like to thank Dr. Bonnie Heck Ferri and Dr. Tucker Balch for serving in my thesis committee.

During my graduate studies, I was supported by the Fulbright Commission in Spain, and I would like to acknowledge them for their financial support and for giving me this extraordinary opportunity. Finally, thanks to all the friends I made these two fantastic years, specially to my girlfriend Pepa Ramirez.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Mobile sensor networks have been shown to be a powerful tool for enabling a number of activities that require recording of environmental parameters at various spatial and temporal distributions. These mobile sensor networks could be implemented using a team of robots, usually called robotic sensor networks.

This type of sensor network involves the coordinated control of multiple robots to achieve specific measurements separated by varied distances. In most formation measurement applications, initialization involves identifying a number of interesting sites to which mobility platforms, instrumented with a variety of sensors, are tasked. This process of determining which instrumented robot should be tasked to which location can be viewed as solving the task allocation problem.

Unfortunately, a centralized approach does not fit in this type of application due to the fault tolerance requirements. Moreover, as the size of the network grows, limitations in bandwidth severely limits the possibility of conveying and using global information. As such, the utilization of decentralized techniques for forming new sensor topologies and configurations is a highly desired quality of robotic sensor networks.

In this thesis, several distributed task allocation algorithms will be explained and compared in different scenarios. They are based on a market approach since our interest is not only to obtain a feasible solution, but also an efficient one. Also, an analysis of the efficiency of those algorithms using probabilistic techniques will be explained.

Finally, the task allocation algorithms will be implemented on a real system consisted of a team of six robots and integrated in a complete robotic system that considers obstacle avoidance and path planning. The results will be validated in both simulations and real experiments.

CHAPTER I

INTRODUCTION

This thesis presents contributions in the field of cooperation within robot teams. More precisely, this research has focused on task allocation applied to formation initialization scenarios.

This chapter firstly presents the motivation and the main objectives of the research carried out. Then, an overview of existing works in the multi-robot task allocation domain is described. Finally, the thesis outline and main contributions are presented.

1.1 Motivation and objectives

Mobile sensor networks have been shown to be a powerful tool for enabling a number of activities that require recording of spatial and temporal variations in environmental parameters required for such activities as monitoring of seismic activity, monitoring of civil and engineering infrastructures, and detection of toxic agents throughout a region of interest [63]. In most sensor network applications, individual sensor agents collect information about their neighboring agents using peer-to-peer communication. Unfortunately, as the size of the network increases, bandwidth limitations and the absence of feasible communication channels severely limits the possibility of using centralized solutions. As such, the utilization of decentralized techniques for forming new sensor topologies and configurations is a highly desired quality of mobile sensor networks.

In future science exploration missions, there is a desire to send multiple, instrumented rovers to scientific sites of interest to expand our understanding of both the history and future of life. Mars exploration missions are focused on finding signs of life to expand our comprehension of where life began. Earth exploration missions are

focused on resolving theories on how life evolved and how it might be effected in the future. These mission examples all have one common theme – human scientists and autonomous rovers must work together to navigate in extreme environments in order to collect scientific measurements of interest. Establishment of these sensor configurations involves determining how to allocate sensor positions to mobile sensor agents in order to achieve a desired topology; a similar research objective that is found when focusing on the task allocation problem with teams of robots.

This thesis proposes different distributed algorithms to solve the task allocation problem applied to mobile robotic sensor networks. Specifically we focus on the problem related to change of formation, i.e., robots have to move from one formation to another, and our algorithms have to ask the question: who goes where? We have named this problem the Initial Formation Problem. The main objective of this thesis is to develop a distributed task allocation algorithm that solves the Initial Formation Problem. Since the tasks consider in this problem are waypoint or navigation tasks, we want an efficient algorithm that obtains solutions close to the optimal in such a way that robots move from one formation to the next one while not wasting valuable energy. Also, our algorithms have to be robust enough to be highly fault tolerant and be able to deal with obstacles, both in the planning and execution levels. Finally, one important objective is to implement our algorithms on a team of real robots to demonstrate the functionality of the system.

1.2 Related work

First, a summary of the literature related with multi-robot task allocation (MRTA) is expounded where different approaches to the problem are explained. Next, an in-depth study of research works that have studied the market-based approach for the multi-robot task allocation problem is presented. Finally, we expound upon a review of the literature related to the Initial Formation Problem.

1.2.1 Different approaches to the multi-robot task allocation problem

The MRTA problem tries to answer the question: which robot should execute each task? Different approaches [1] have been developed in the last decades. An initial classification can be made at the organization level: centralized, distributed and hybrid. Next, we will comment on the main research threats found in each one.

1.2.1.1 Centralized approach

In a centralized approach, all the information is transmitted to a central server that usually calculates the optimal allocation. This approach is less robust than the distributed one, mainly because of the existence of a central element ([13] and [15]). Furthermore, this element must have enough computing capacity to calculate an optimal or at least suboptimal allocation in a coherent time since the MRTA problem is $\mathcal{NP}$-hard [46]. This statement can be proven since the different MRTA problems resemble the Traveling Salesman Problem [49], the Min-Max Vehicle Routing Problem [4] and the Traveling Repairperson Problem [38] which are intractable even on the Euclidean plane. For this reason, the centralized approach uses results from the Operations Research field of study, mainly the ones related to the Traveling Salesman Problem. Other problems related to this approach includes limitations with communication coverage, robustness and scalability. The primary advantage is that solutions are usually optimal or very close to the optimal, but always under the assumption that the information from the different robots is accurate enough.

1.2.1.2 Distributed approach

In a distributed approach, robots use local information and inter-agent communication to allocate the different tasks without the need of a central element. The

[1] Although the task allocation problem has been studied in the field of multi-agent systems, usually those algorithms cannot be applied to multi-robot systems. The main reason is that multi-agent systems do not consider the uncertainties that are so important in robotics. This the reason is why these research works are not going to be considered in this chapter.

distributed approach is more complex because robots have to allocate tasks by their own. If the system is truly distributed, they only have access to local information. One of the few truly distributed solutions is ALLIANCE [61] based on distributed behaviors that uses motivations mathematically modeled. Task allocation is based on the implementation of motivations, which are impatience and consent. Impatience allows a robot to manage a situation where other robots fail to execute their own tasks. Meanwhile, consent allows a robot to manage a situation where it fails to execute its own task. These motivations activate the different behaviors using information about the environment and the rest of robots. For example, from the time a new task is announced, the level of impatience of each robot increases at different speeds. The more suitable the task is for the robot, the faster the level of impatience increases. When impatience exceeds a limit, the robot activates the behaviors needed to execute the task. From that moment, the impatience of the rest of the robots increases much slower to avoid having more than one robot executing the same task. This algorithm was improved with learning techniques used for parameter adaptivity in [60].

A similar idea is developed in [1] and [42] called threshold-based task allocation where each robot has an activation threshold for each task that needs to be performed. They define stimulus as a value that reflects the urgency or importance of performing a task. The stimuli are perceived continuously for each of the tasks. When the stimulus for a robot exceeds a certain threshold, it starts executing the task. When the stimulus falls back below the threshold, the robot stops the behaviors that execute the task. This reaction to the stimulus can be deterministic or probabilistic.

Another solution based on behaviors can be found in [79], where a system called BLE is used to resolve the CMOMMT (*Cooperative Multi-robot Observation of Multiple Moving Targets*) problem. This system extends the subsumption architecture [12] to multiple robots. To achieve this, BLE uses suppression and inhibition techniques between the behaviors of the different robots of the system (cross inhibition).

Basically, it is based on the fact that each robot executes the task that it is better prepared for, in comparison with the rest of the robots. At the time a robot starts executing a task, it inhibits the same level behaviors from the rest of robots. In that way the robot is demanding the task. Cross inhibition is an active process. If one robot fails, it will stop to inhibit the behaviors of the other robots, and afterwards, one of them will take care of the task.

Those systems based on behaviors have high fault tolerance, and also, adapt very well to noisy environments. However, those architectures do not take into consideration the efficiency of the solution and their only aim is to finish the mission successfully. We define a mission as a partially ordered group of tasks. Most of the rest of the algorithms fall within the domain of what we call *time-distributed* algorithms, which means that robots make decisions based on inter-agent communications transmitted at different time instances. This type of algorithm is more fault tolerant than a centralized approach, and can obtain more efficient solutions than a completely distributed approach.

Within the *time-distributed* approach, negotiation techniques based on market rules have been developed. These techniques have received significant attention [20], since they offer a good compromise between communication requirements and the quality of the solution. In the last decade, several projects have used this approach [10] and [29]. Some of these works [16], [70] and [78] take into consideration both the success of the tasks as well as the efficiency of the solution. These techniques will be commented deeply in the next section. Another technique, called token-based [23] and [71], can be considered also within the *time-distributed* approach. In this technique each token represents a different task to be executed. When a robot receives a token, it decides whether to perform the task associated to it or to pass the token onto another agent. This decision is usually made based only on local information. To prevent conflicts regarding token coherence, distributed algorithms have also been

developed [22]. Another work with a similar approach can be found in [82] where the robots use local decision theoretic models to determine when and where to pass the tokens.

In [26], a different task allocation algorithm is described within the *time-distributed* approach. Instead of using utilities or costs to compare different tasks, they use a suitability metric. This allows them to transform the task allocation problem into a Transportation Problem (TP) [56] which has $\mathcal{P}$ complexity, instead of treating the problem as a Multiple Traveling Salesman Problem or similar ones that are $\mathcal{NP}$-hard. However, their algorithm maximizes the robot suitability for each task which does not mean, for most of the cases, that the utility will be as well maximized. In this approach robots exchange task suitability matrices (TSM) that models the state of the system. Using the updated information, each robot solves the TP problem. Nevertheless, they do not consider situations when the information is incorrect or the robots have different situational awareness of the system, as will be considered in the next algorithms. Finally, they consider the fault tolerance aspect [25] using a backoff scheme which is based on ideas from communication protocols.

In the case that we know all tasks and robot states at the beginning, a valid distributed approach could be to run an optimal algorithm on each robot, and in theory, all the robots will obtain the same solution. Each robot will execute the tasks that the optimal algorithm has decided, and they will not overlap. However this method strongly depends on the assumption that information is accurate and all the robots have the same situational awareness, which is not always true in a distributed robotic system. Therefore, approaches are needed that are robust to differences in the situational awareness of each robot and inaccurate information. In [31], a distributed robust approach where all the robots have a common cooperative scheduling strategy is explained. They make use of exclusion algorithms to deal with the asynchronous aspect of the decision making process and the inaccuracy of the information.

A different work, within the same approach, can be consulted in [2] that uses an algorithm that adds a second planning step based on sharing the planning data. This approach is analogous to closing a synchronization loop on the planning process to reduce the sensitivity to incorrect data. This work has been applied to the task allocation problem for a group of Unmanned Aerial Vehicles (UAVs) [9]. They use consensus techniques to overcome the differences between the situational awareness of each UAV. A new version of the algorithm can be found in [3] that combines robust planning with techniques developed to eliminate the churning coming from the replanning when the situational awareness is updated. This version is less conservative than robust planning and does not suffer from churning type of instability. Another work that uses a similar approach is [7]. In this case, every robot calculates the optimal solution assuming perfect global information. Afterwards, a consensus algorithm is used to ensure that each robot has consistent information. The consensus algorithm uses the communication network, but only local communication is assumed. The approach is implemented on a team of fixed-wing UAVs. Finally, in [5] mixed integer linear programming techniques with local information is used for task allocation. In order to improve the results, information is shared between neighbors. Although only local information is used for the linear programming algorithm, they state that the system will reach an equilibrium when all the robots find the same solution [6].

On the other hand, we can find another group of truly distributed task allocation algorithms. However, they do not allocate specific tasks to robots, their only aim is to divide the group of robots in subgroups, where each subgroup will execute a different task. For example, we might want to have 30% of the robots executing task A and the other 70% executing task B. In [51] a dynamic task allocation algorithm for this type of problem with theoretical analysis is presented. Also, in [53] four different algorithms are explained and tested with real robots. Although this type of problem is considered within the multi-robot task allocation domain, it will not be studied in

the present work.

1.2.1.3 Hybrid approach

A hybrid approach tries to improve the efficiency of the solution with respect to the distributed systems without reducing too much the fault tolerance aspect. The common solutions use several central elements or dynamic clusters of robots ([16] and [41]).

1.2.1.4 Comparisons

Comparison between different approaches are difficult since there is no common set of missions, and also, it is not easy to implement algorithms developed by others. However, there are some works that compare different approaches, for example in [17] they compare different task allocation methods, both centralized and distributed (behavior and market-based approaches). They conclude that the market-based approach is the best option since it offers a good compromise between communication requirements and the quality of the solution. Also, the market-based approach is compared with the threshold-based approach in [40] and with the token-based approach in [81]. They conclude that the market-based approach obtains more efficient solutions but it usually needs more communication requirements. Also in [40], it is pointed out that market-based approaches need accurate information about tasks and local states to work properly. When the information is not accurate, threshold-based approaches offer the same quality of the solution with a fraction of the requirements. A comparison between different market-based algorithms and different optimization criteria can be consulted in [55]. As can be seen, the market-based approach has received a lot of attention and it has been used as a baseline to prove whether a new approach is good enough.

1.2.2 Market based approach to the task allocation problem

The market-based approach [20] is founded on the *Contract Net Protocol* or CNP [65, 72]. The main idea of the CNP is to allocate tasks through negotiation of contracts. An agent "director" may offer a task to the rest of the agents "contractors". Afterwards, they must announce their bids based on their capacity to execute that task. Next, the "director" applies a mechanism that awards the task to one of the "contractors" and finally the winner executes the task. In principle, agents are not designed to be "directors" or "contractors", instead those roles are played dynamically. The mechanism usually used for the allocation is based on auctions where tasks are awarded to the robot with the best bid.

Likely, the first distributed market based system was M+ [10], defined within a general architecture for the cooperation among multiple robots [11]. In this system robots have a local plan of at most two tasks. For that reason, when a robot calculates the task cost, it considers the next one in order to increase the efficiency of the solution.

Another similar system is MURDOCH ([28] and [29]). In this system tasks are only allocated to robots that are idle, i.e., during the execution of a task, robots do not take part in the different auctions. Therefore, the mechanism of task allocation is based on a purely greedy method (the best solution at a certain time only considering the state of the system in that moment). The results obtained with this algorithm are less efficient than M+, because it does not consider future tasks that robots plan to execute. However, the main advantage of this method is that it is very simple and needs few resources.

TraderBots [18] is a distributed system whose main contribution is the consideration of dynamic environments [19], partial failures of the robots and communications. At the same time it obtains efficient solutions. Unlike the other systems commented above, robots have a local plan and more than one task can be allocated to each one.

For this reason, it obtains efficient solutions. Moreover, the efficiency is improved even more [16] using two different techniques:

1. The use of leaders in clusters of robots that allows negotiations among more than two robots at the same time.

2. The capacity to negotiate group of tasks, instead of negotiating only one task as occurs in M+ and MURDOCH.

After showing various results of simulations [16] using these techniques, it is concluded that a greater efficiency is achieved when it is possible to negotiate more than one task at a time rather than negotiate more than one robot at the same time using leaders.

Another interesting system is the so called *Distributed and Efficient MultiRobot Cooperation Framework* (DEMIR-CF) [66], [67] and [68]. The assignment of tasks is done incrementally, so robots only consider one task per time and they do not have an internal plan. However, they maintain a list of possible tasks to be executed and they select from it which task is going to be used in the next auction. The list of tasks is obtained based on heuristic algorithms using costs. They also consider tasks that need more than one robot to be executed. To solve this problem, they create dynamic coalitions with a leader and it is the leader who is in charge of selecting which robots will form the coalition. Also the leaders release robots from the coalition when it is necessary. Finally, precautions routines are used to prevent inconsistencies in the system due to robot or communication failures [69]. This framework has been successfully applied to naval mine countermeasure missions [70].

The projects commented above are medium-large projects. However, there are other short works that show new and really interesting characteristics. In [33] a market system based on the exchange of tasks is presented. The main advantage to this approach is that it is not necessary to have a common metric among the values

of the different robots' cost functions. This allows robots to calculate the cost of each task based on their own parameters and it is not necessary to normalize them with the rest of robots. Usually, all projects use cost or utility functions that are really simple, and therefore, they are easy to normalize with other robots. However, if very different tasks were considered, the normalization step would not be so easy, and the need to have a common metric could be a problem. The objection of this system is that it needs a large number of interactions to achieve an exchange of tasks, since it is only produced when the exchange is beneficial for both robots, which is not known a priori. In [34], an algorithm was developed inspired by the market and threshold approaches, that tries to determine the optimal number of robots needed to solve specific tasks. Finally, [57] presents some variations of the CNP protocol in such a way that a mission could be completed even if mobile objects hinder the execution of tasks. This system is based on the reallocation concept and, due to the dynamism of the environment, every time a robot finishes one task, it announces again all the planned tasks that are not in execution.

1.2.3 The Initial Formation Problem

The Initial Formation Problem [36], is a rendition of the MRTA problem, in which each robot can only be allocated to one task. In order to illustrate the differences between both problems, we can think about the general task allocation problem as a Multiple Traveling Salesman Problem [49, 47] and the Initial Formation Problem can be viewed as a classical job assignment problem [43] where robots are the workers and tasks are the jobs to be executed by those workers.

The Initial Formation Problem has received less attention in the task allocation domain than the general problem. However, this type of problem becomes important within the field of formation control ([37, 50]) where using local information and control laws, the distributed algorithm is able to drive a given formation error to

zero. As it is stated in [39], these algorithms require a first step that assigns the robots to the formation positions while taking into account their initial positions, i.e., answer the question, "who goes where?" In the robot cooperation domain, the Initial Formation Problem, cast as a basis algorithm, could play the same role that the job assignment problem played in the Operations Research field. A better understanding of the Initial Formation Problem could lead to improvements in solutions for more difficult problems, such as the general task allocation problem.

Different approaches can be used to solve the Initial Formation Problem. This problem has typically been solved optimally using centralized solutions such as the Hungarian method [43]. The computational complexity of this problem is not the main issue since the Hungarian method has complexity $\mathcal{O}(n^3)$ (where n is number of robots or tasks), and there are algorithms with complexity $\mathcal{O}(n^2)$ [75]. However, this kind of solution assumes that all the information is available and has the disadvantages related to centralized systems: low fault tolerance and slow response to dynamic changes in the environment. Other approaches, such as [8], use a parallel algorithm based on auctions to obtain the optimal solution for the assignment problem. However, a parallel algorithm needs to have updated information transmitted between all the nodes. From a robotics perspective, this approach does not lead to any advantages in fault tolerance or response to changes in the environment, since it only improves the time complexity of the algorithm. A modification of this algorithm for mobile agents can be found in [54], but still they need to maintain a complete knowledge of task prices. Different distributed approaches have also been developed recently, but tasks have to be communicated to all the robots at the beginning. A distributed heuristic algorithm with local communication is explained in [83], while in [84] the agents are controlled by hybrid models using distributed potential fields. Both approaches fail to return a highly efficient solution since more than one robot can execute the same task. In [73], a solver of the Traveling Salesman Problem (TSP)

is used to decide which robot should execute which task. However, this approach first solves a much more difficult problem to obtain a solution to the assignment problem.

In this thesis, we have decided to use market-based algorithms to address the Initial Formation Problem [36]. This approach offers a good compromise between communication requirements and the quality of the solution. It is an intermediate solution between centralized systems where all the information is available, and distributed systems, where only local information is accessible. Moreover, this approach obtains solutions close to the optimum and offers a good level of fault tolerance and reaction to changes in the environment.

1.3 Limiting the scope

In this section, we explain the characteristics and assumptions considering in the multi-robot system and task allocation problem for this thesis.

1.3.1 Multi-robot system

First, we classify the multi-robot system considered in this thesis in the system dimensions, using the taxonomy presented in [21], as:

- Collective size: the number of robots considered is limited, although the number of robots could be in the range of tens of robots (SITE-LIM).
- Communication range: initially we suppose that robots can communicate with any other robot (COM-INF). But, the algorithms presented in this thesis will work with limited communication range. Although, as a logical consequence, the quality of the solutions will decrease.
- Communication topology: robots are linked in a general graph (TOP-GRAPH).
- Communication bandwidth: the cost of communication is negligible compared to other costs (BAND-INF).

- Collective reconfigurability: the relationship among robots can be reconfigured dynamically (ARR-DYN).

- Processing ability: each robot can be thought of as Turing Machine Equivalent (PROC-TME).

- Collective composition: we have only tested our algorithms in systems where robots have the same behavioral level (CMP-HOM). However, there is no reason why our algorithms cannot be applied to a heterogeneous team of robots.

On the other hand, the considered multi-robot system can be classified in the coordination dimensions, using the classification stated in [24], as:

- Cooperation level: our system is considered cooperative since we suppose that robots operate together for the benefit of the team. We do not consider situations where robots aim to behave against the global goal deliberately.

- Knowledge level: our system is aware since robots have some knowledge of their team mates.

- Coordination level: the actions performed by each robot takes into account the actions executed by the other robots as a result of the coordination protocol. Therefore, it can be said that our system has a strong coordination.

- Organization level: since our system does not consider the existence of leaders and there is no central element, our system is distributed.

1.3.2 Multi-robot task allocation problem

As was said before, our objective is to solve the Initial Formation Problem in a distributed way. Although the problem has already been described, a more formal definition can be stated as follows

Given a number of tasks, $\{T_1, T_2, ...T_T\}$*, a team of robots* $\{R_1, R_2, ..., R_R\}$*, and a*

function $U(T_i, R_j)$ that specifies the utility of executing task T_i by robot R_j, find the assignment that allocates one task per robot and tries to optimize some predefined metric.

In order to define completely the Initial Formation Problem, we need to specify the utility function and the metric that needs to be optimized. Usually, the utility is composed of the reward and the cost functions.

$$U(T_i, R_j) = R(T_i, R_j) - C(T_i, R_j)$$

The reward function indicates which is the benefit of executing a task, and the cost functions gives you an estimate of the effort to accomplish the same task. In this thesis, we do not consider rewards associated to the tasks, so the utility functions are equal to the cost of the tasks.

On the other hand, different metrics [76] can be considered such as: the sum of the utilities, the maximum of the utilities and the average of the utilities associated to the tasks executed by the different robots. In this thesis, our metric is the sum of all the utilities. Since only costs are considered, our objective is to minimize the sum of the costs. Therefore, the version of the Initial Formation Problem that is used along the thesis can be stated as

Given a number of tasks, $\{T_1, T_2, ...T_T\}$, a team of robots $\{R_1, R_2, ..., R_R\}$, and a function $C(T_i, R_j)$ that specifies the cost of executing task T_i by robot R_j, find the assignment that allocates one task per robot and tries to minimize the global cost defined as $\sum_{j=1}^{M} C(T_i, R_j)$, where task i is assigned to robot j.

Since we are working with formations, the cost is defined as a quantity that reflects how much it will cost the robot to go to a certain waypoint, such as the traveled euclidean distance or the traversability index [35]. However, the algorithms presented in this thesis could be used with other costs functions such as energy or time and with other types of tasks different than waypoint tasks.

Using the concepts from the MRTA formal taxonomy explained in [30], the Initial Formation Problem considered in this thesis has the following characteristics:

- Single-task robots(ST): each robot is capable of executing as most one task at a time.
- Single-robot tasks(SR): each tasks required exactly one robot to achieve it.
- Time-extended assignment (TA): tasks are not allocated instantaneous, and for some of our algorithms more information is available to take the decision. The only exception is the BS-WR algorithm that belongs to the instantaneous assignment category. This algorithm is explained in detail in Section 2.1.

Thus, our problem can be designated as ST-SR-TA.

1.4 Thesis outline

The thesis consists of five chapters. A summary of the contents of the chapters is presented here:

In Chapter 2, different distributed task allocation algorithms founded on a market-based approach are explained. First, two basic algorithms are described. The importance of the use of reallocations is pointed out. Next, three new improved algorithms are shown. These algorithms obtain solutions closer to the optimal with a small increase in the shared information. All the algorithms have been tested in simulation in two different scenarios. The two most interesting algorithms have been also tested with real robots and the results are similar to the ones obtained from simulations. Finally, it is stated that our algorithms do not need any synchronization mechanism. Simulations with and without synchronization have been run to prove this fact.

Chapter 3 presents a general probabilistic analysis that is used to compare different market-based task allocation algorithms applied to the Initial Formation Problem. This analysis is used to calculate the expected value of the global cost. This metric

gives us an idea of the behavior of the algorithm over time. The analysis is developed for two of our algorithms but it could easily be extended to the rest. Also, our analysis can handle situations when the number of robots and tasks are different. The results are validated for two different scenarios with simulations and experiments with real robots.

In Chapter 4 considerations for real world applications are studied. First, situations with large number of robots are considered. These situations can saturate the network due to the large number of messages. An adaptive algorithm has been developed to reduce the number of messages. Next, a distributed fault tolerance algorithm is explained. This algorithm is able to recover the assigned tasks from robots with failures even when these failures affect the communication capabilities of the robot. The task allocation algorithms are integrated within a complete robot architecture. This integration involves the communication between the task allocation and path planning modules in order to calculate realistic costs when obstacles are considered. This integration is tested both with simulations and real experiments. Finally, it is explained how the presented work is applied to mobile sensor networks for achieving scientific measurements in arctic environments.

The thesis is completed with Chapter 5, which discusses and concludes the results of the thesis and in which the future work is summarized.

CHAPTER II

MARKET-BASED APPROACHES APPLIED TO THE INITIAL FORMATION PROBLEM

In this chapter a market-based approach will be used for addressing the Initial Formation Problem. The main reason for using this approach is that we are interested in not only obtaining a feasible solution, but also an efficient one. Also, this approach offers a good compromise between communication requirements and the quality of the solution. It is an intermediate solution between centralized where all the information is available, and distributed where only local information is accessible, while obtaining solutions close to the optimum and offering a good level of fault tolerance and reaction to changes in the environment.

A market-based algorithm is typically implemented by using some variant of the *Contract Net Protocol* [65, 72], where two roles are played dynamically by robots: auctioneer and bidders. The auctioneer is the robot in charge of announcing the tasks and selecting the best bid from all the bids received from the bidders. The best bid is considered the one with the lowest cost.

First, two basic algorithms are explained to introduce the concepts related to market-based algorithms and show the importance of reallocations in the quality of the solution. Then, different modifications of the basic market-based algorithm that improve its results will be addressed. All the algorithms are tested in simulation and the results are commented. Also, two of the most important algorithms will be tested with real robots. Finally, it is demonstrated that these algorithms do not need any type of synchronization method.

2.1 BS-WR: BaSic market-based algorithm Without Reallocations

In this algorithm, bidders broadcast their bids only if they do not already have an assigned task, i.e., when a task is allocated to a robot, it no longer bids on other tasks in the auction. This algorithm is easy to implement and uses a small number of messages. However, the solution depends on the order that tasks are announced and, as such, may not result in an efficient solution. The bidder and auctioneer algorithms are explained in Algorithms 1 and 2.

Algorithm 1 Bidder algorithm

```
a new message is received
if new message is a task announcement then
  if won-task list is empty then
    calculate bid (distance to the task)
    send bid to the auctioneer
  end if
else if new message is a task award then
  introduce won task in the won-task list
end if
```

Algorithm 2 Auctioneer algorithm

```
if announcement task is not empty then
  announce task
  while timer is running do
    receive bids
  end while
  calculate best bid
  send task to best bidder
  delete task from announcement list
end if
```

2.2 BS: BaSic market-based approach

The basic idea of this algorithm is that each robot must have only one task, so it will keep the task with the lowest cost. The auctioneer and bidder algorithms are explained in Algorithm 3 and 4 respectively. If a robot wins a new task that has a

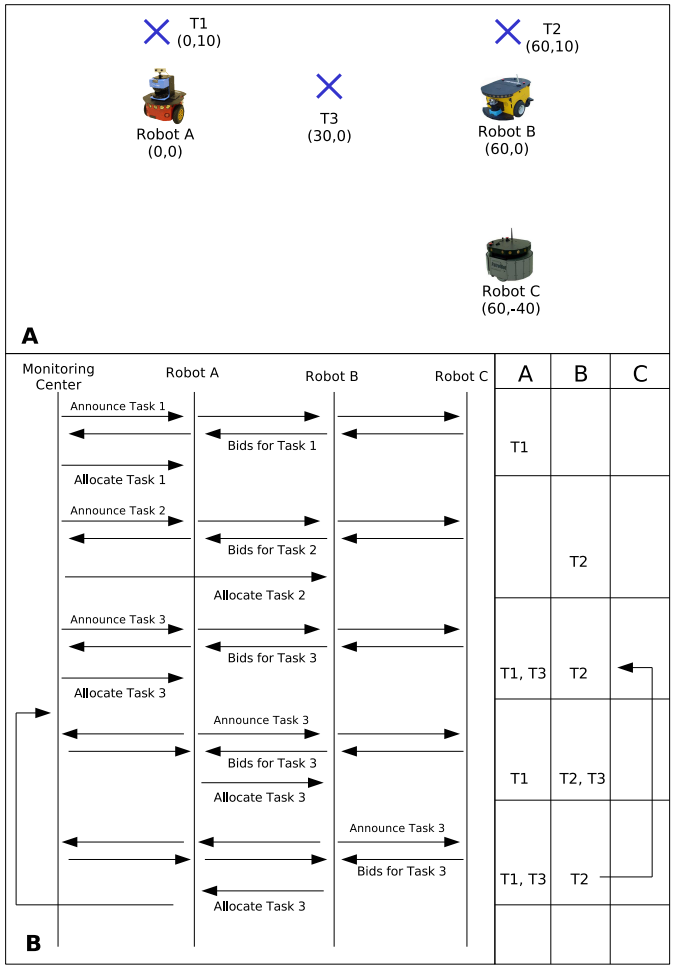


Figure 1: Figure A presents the initial position of the robots and the tasks. Figure B presents the messages exchanged among the different agents and shows how an infinite loop appears in the negotiation protocol.

lower cost than the one already won, it will sell the old task to the robot with the best bid but worse than its own bid. The best bid worse than the robot's bid is selected in order to avoid infinite loops in the negotiation. This scenario could happen when two robots have the best bids for at least three tasks as shown in Figure 1.

From the results shown in Section 2.4, it can be stated that this algorithm obtains satisfactory results when the initial position of the robots and the tasks are calculated totally at random. However, there are situations when this algorithm does not obtain good results which usually happens when a robot has to execute a task that is the worst one for its own interest, as can be seen in Figure 2. In this example, the global

Algorithm 3 Auctioneer algorithm

```
if announcement-task list is not empty then
  announce task
  while timer is running do
    receive bids
  end while
  calculate best bid worse than the robot's bid
  send task to best bidder
  delete task from announcement-task list
end if
```

Algorithm 4 Bidder algorithm

```
a new message is received
if new message is a task announcement then
  calculate bid (distance to the task)
  send bid to the auctioneer
else if new message is a task award then
  if the robot has already won a task then
    if cost of the new task < cost of the won one then
      introduce old task in announcement-task list and delete it from won-tasks
      list
      introduce won task in the won-tasks list
    else
      introduce won task in the announcement-task list
    end if
  else
    introduce won task in the won-tasks list
  end if
end if
```

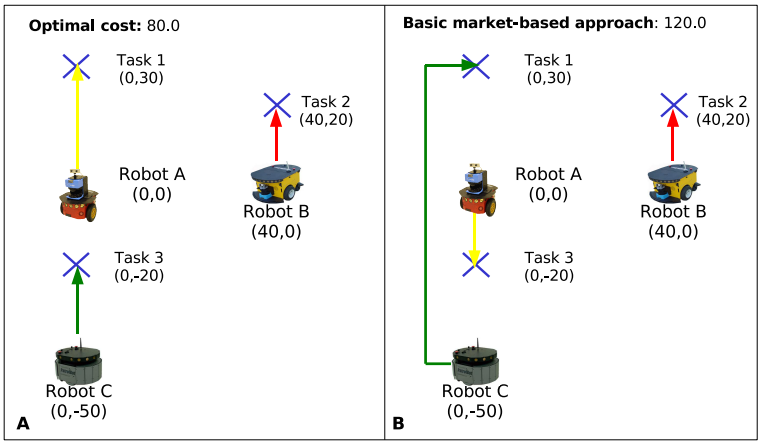


Figure 2: Difference in cost between the optimal allocation and the one obtained with the basic market-based algorithm.

cost obtained with the market-based algorithm is 66.67% greater than the optimal allocation.

2.3 Improved Algorithms

In order to solve the initial formation problem, the task allocation algorithm has to solve two main problems:

- How do I calculate the bid for a certain task?
- If I won more than one task, how do I determine which one to keep?

In the BS algorithm, bids are the distance between the robot position and the tasks (we are only considering waypoint tasks) and if one robot wins more than one task, it keeps the one that is closest to itself, i.e., the one with the lowest cost or best bid. Therefore, if our objective is to improve the BS algorithm, one or both of these aspects must change. Moreover, the improved algorithm must keep the advantages of the market-based approach: fault tolerance, independent from the number of robots and high adaptation to changes in the environment using reallocations.

2.3.1 RMA: Robot Mean Allocation algorithm

Our first improved algorithm (RMA) is focused on trying to choose in a more clever way the task that must be kept when a robot wins more than one task. This is accomplished using additional knowledge available to the system. Instead of keeping the task with the smallest distance to the robot, the task with the highest difference between the distance to the robot and the mean of its distance to all the robots will be selected. In other words, suppose that there are a finite number of robots N_R and robot R_k has won tasks T_i and T_j. In this case, robot R_k will keep task T_i if and only if

$$\sum_{r=1}^{N_R} \frac{D(R_r, T_i)}{N_R} - D(R_k, T_i) > \sum_{r=1}^{N_R} \frac{D(R_r, T_j)}{N_R} - D(R_k, T_j)$$

where $D(R_k, T_i)$ is the distance between robot R_k and task T_i. The reason behind this idea is to make the robot willing to choose the task that is best for the team, not just for itself. So, robots will more probably win tasks that have a high cost for the rest of the robots and a low one for themselves.

The question that arises now is how to calculate the mean of the distances for a certain task. During the normal operation of the algorithm, the auctioneer receives bids from all functioning robots in order to allocate the task to the best robot. At this moment, the auctioneer knows all the distances between every robot and the current task. Thus, the mean is calculated by the auctioneer and transmitted to the robot within the message that informs the robot that has won the task. The major difference with the BS algorithm is that the robot must remember the mean associated with the won task. Furthermore, the robot is able to compare their means to different tasks because it remembers the mean of the task already won and the mean of the new allocated task is sent by the auctioneer, as was explained previously.

2.3.2 TMA: Task Mean Allocation algorithm

In the TMA algorithm, instead of changing the way that tasks are selected, the cost function will be changed. In the original algorithm the cost function used to calculate the bid for a certain task is the distance between the robot and the task. However, in this improved algorithm the cost function will be the difference between the distance of the robot and the task minus the mean of the distances between that robot and all the tasks, i.e.,

$$C(R_i, T_j) = D(R_i, T_j) - \sum_{t=1}^{N_T} \frac{D(R_i, T_t)}{N_T} \tag{1}$$

where $C(R_i, T_j)$ is the cost function for robot R_i and task T_j and the total number of tasks is N_T. The idea is to decrease substantially the cost of a task when is close to a robot and the rest of the tasks are far away from the same robot. But if all the tasks have similar costs, the cost reduction will be smaller. Therefore when bids are received by the auctioneer, the tasks from robots that are in the first situation will be favor with respect to the tasks from robots that are in the second situation. Also, if a task is far away from a robot and the rest are close to the robot, the cost of the task will be kept almost the same.

The rest of the algorithm works the same as the BS algorithm but using the new cost function instead of the distance. At the bidder side, when one robot wins two tasks instead of comparing the distances to choose the closest one, it will compare the costs using the new cost function and it will select the task with the lowest cost for itself. Thus, a robot R_k that have won two tasks, T_i and T_j will keep T_i if and only if

$$C(R_k, T_i) < C(R_k, T_j),$$

or

$$D(R_k, T_i) - \sum_{t=1}^{N_T} \frac{D(R_k, T_t)}{N_T} < D(R_k, T_j) - \sum_{t=1}^{N_T} \frac{D(R_k, T_t)}{N_T}. \tag{2}$$

As it can be seen in Equation 2, the sum factor is equal in both parts of the inequality.

So, the tasks selected at the bidder side of the algorithm will be the same way by using either the distance or the new cost function.

The differences between the BS and TMA algorithm appears at the auctioneer part of the algorithm. When the auctioneer receives the costs from all the bidders, it will no longer receive information about the distances between the robots and the tasks. It will receive the new costs (see Formula 1) where the right part of the formula will be different for each robot, i.e., an auctioneer will assign task T_i to robot R_j if an only if

$$D(R_j, T_i) - \sum_{t=1}^{N_T} \frac{D(R_j, T_t)}{N_T} < D(R_k, T_i) - \sum_{t=1}^{N_T} \frac{D(R_k, T_t)}{N_T}, \forall k \neq j.$$

The only drawback to this improvement is that robots must know the different tasks at the beginning in order to calculate the mean of the distances. However, the extra resources needed for the algorithm are almost the same as that for the BS algorithm since robots only have to memorize the mean calculated at the beginning and implement one basic cost function operation.

2.3.3 RTMA: Robot and Task Mean Allocation algorithm

The last algorithm is a combination between the RMA and the TMA algorithms. Therefore, the cost function will be the one used in the TMA algorithm, while the logic used to select tasks is the one used in the RMA algorithm. Just as in the TMA algorithm, the same results are obtained whether the distance or the new cost function is used in the logic that selects tasks at the bidder part of the algorithm.

First, if the distances are used, task T_i will be the one selected if and only if

$$\sum_{r=1}^{N_R} \frac{D(R_r, T_i)}{N_R} - D(R_k, T_i) > \sum_{r=1}^{N_R} \frac{D(R_r, T_j)}{N_R} - D(R_k, T_j). \quad (3)$$

On the other hand, if the new cost function is used, task T_i will be the one selected if and only if

$$\sum_{r=1}^{N_R} \frac{C(R_r, T_i)}{N_R} - C(R_k, T_i) > \sum_{r=1}^{N_R} \frac{C(R_r, T_j)}{N_R} - C(R_k, T_j)$$

where $C(R_r, T_i) = D(R_r, T_i) - \sum_{t=1}^{N_T} \frac{D(R_r, T_t)}{N_T}$.

Thus,

$$\sum_{r=1}^{N_R} \frac{D(R_r, T_i)}{N_R} - \sum_{r=1}^{N_R}\sum_{t=1}^{N_T} \frac{D(R_r, T_t)}{N_T \cdot N_R} - D(R_k, T_i) + \sum_{t=1}^{N_T} \frac{D(R_k, T_t)}{N_T} > \sum_{r=1}^{N_R} \frac{D(R_r, T_j)}{N_R} - \sum_{r=1}^{N_T}\sum_{t=1}^{N_T} \frac{D(R_r, T_t)}{N_T \cdot N_R} - D(R_k, T_j) + \sum_{t=1}^{N_T} \frac{D(R_k, T_t)}{N_T}.$$

Simplifying the previous equation,

$$\sum_{r=1}^{N_R} \frac{D(R_r, T_i)}{N_R} - D(R_k, T_i) > \sum_{r=1}^{N_R} \frac{D(R_r, T_j)}{N_R} - D(R_k, T_j). \quad (4)$$

As can be seen, Equations 3 and 4 are exactly the same. However, due to practical implementation, it is easier to compare the tasks using the new cost function since the bids are calculated with it.

The auctioneer algorithm is practically the same as shown in Algorithm 3. The best bid is still the one with the lowest cost, but the bid is calculated with the new cost function. On the other hand, the new bidder algorithm is explained in Algorithm 5.

2.4 Simulations and Discussion

A multi-robot simulator has been used to test decentralized algorithms. This simulator is based on an architecture designed for heterogeneous robots [78] and divided into three layers. The highest layer is independent from the type of robot and is the one aware of the existence of other robots. Thus, the task allocation algorithm is implemented in this layer. Moreover, the communication among robots is based on IP, so it can also be used as an interprocess communication method for simulations. The other two layers are used to execute the different tasks allocated to the robot and makes the creation of new algorithms easier by using a modular and component-based architecture.

Algorithm 5 Bidder algorithm

```
a new message is received
if new message is a task notification then
  mean = mean + D(R_k, T_i)/N
else if new message is a task announcement then
  calculate bid (distance to the task minus mean)
  send bid to the auctioneer
else if new message is a task award then
  if the robot has already won a task then
    if cost of the new task - mean of the costs > cost of the won one - mean of
    the costs then
      introduce old task in announcement-task list and delete it from won-tasks
      list
      introduce won task in the won-tasks list
    else
      introduce won task in the announcement-task list
    end if
  else
    introduce won task in the won-tasks list
  end if
end if
```

The different algorithms have been tested using initial positions of the robots and formations calculated at random in a virtual world of 1000x1000 meters. First, the BS-WR algorithm is compared against the BS algorithm to state the advantage of the use of reallocations in increasing the performance of the task allocation algorithm. As can be seen in Figure 3, the BS algorithm obtains better results than the BS-WR algorithm for all the cases. Also, when reallocations are used, the final allocation does not depend on the order of the task announcements.

Next, all the algorithms except the BS-WR algorithm have been simulated using a variety of scenarios in which the number of robots and tasks ranged from 2 up to 20, and for every case one hundred simulations were run. These results are shown in Table 1 where, in each cell, the mean of the global cost and the error in percentage in comparison with the optimal solution are presented. The optimal solution has been calculated using the Hungarian method [43]. In order to show the results clearly,

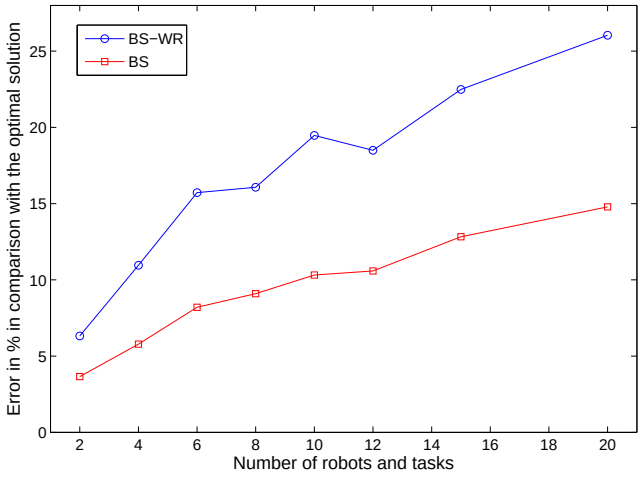


Figure 3: Mean error in percentage in comparison with the optimal solution for the BS and BS-WR algorithms where the inital positions of the robots and the points of the formations are calculated at random over 100 simulations.

only the error in percentage is shown in Figure 4. It can be observed that the best algorithm is the RTMA and the worst one is the basic market-based (BS) algorithm, although all the algorithms obtains efficient results up to 8 robots and tasks where the largest error is less than 10%. For more than 8 robots, only the RTMA algorithm obtained good results, with a maximum error of 5.98% in the case of 20 robots. As can be seen in Figure 4, the error with the optimal solution increases linearly for all the algorithms with respect to the number of robots and tasks. However, the RTMA algorithm is the one with lowest slope. Furthermore, it is interesting to comment that with less than 10 robots the RMA algorithm obtains results slightly better than the TMA algorithm, but with over 10 robots the TMA algorithm obtains better results than the RMA. It is also important to point out that for 2 robots and tasks the RMA and RTMA algorithms always obtain the optimal solution.

The results of Table 1 only show statisically how good the algorithm is based on the mean. However, it could be the case that an algorithm could have good results on average but there are some situations where its results have large errors. Therefore, another important parameter to consider is the maximum error with respect to the

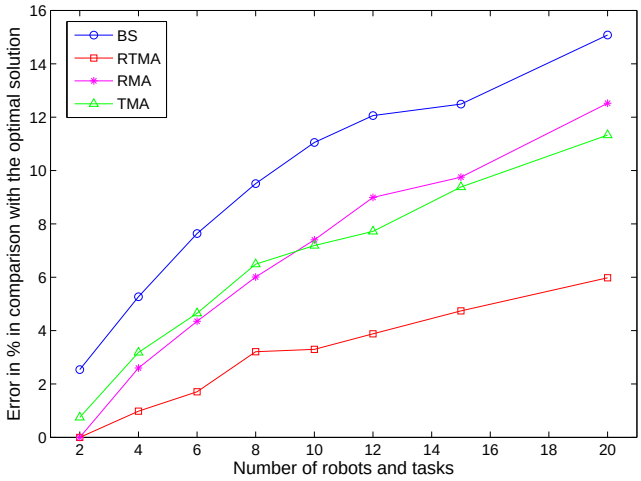


Figure 4: Mean error in percentage in comparison with the optimal solution for the different types of algorithms and calculating the inital positions of the robots and the points of the formations at random over 100 simulations.

Table 1: Results computed for formations with different number of robots and tasks over 100 simulations per each case. In each cell the mean of the global cost and the mean error in percentage with the optimal solution are presented.

Tasks & Robots	***BS***	***TMA***	***RMA***	***RTMA***	***Optimum***
2	909.35	893.48	886.84	886.84	886.84
	(2.54%)	(0.75%)	(0.0%)	(0.0%)	
4	1473.52	1444.22	1436, 16	1413.45	1399.73
	(5.27%)	(3.18%)	(2.6%)	(0.98%)	
6	2020.13	1964.07	1958.49	1908.85	1876.77
	(7.64%)	(4.65%)	(4.35%)	(1.71%)	
8	2443.57	2376.03	2365.30	2302.90	2231.27
	(9.51%)	(6.49%)	(6.01%)	(3.21%)	
10	2865.81	2766.18	2771.63	2666.06	2580.65
	(11.05%)	(7.19%)	(7.4%)	(3.30%)	
12	3233.25	3108.09	3144.98	2997.34	2885.35
	(12.06%)	(7.72%)	(8.99%)	(3.88%)	
15	3749.97	3646.21	3658.48	3491.44	3333.55
	(12.49%)	(9.38%)	(9.75%)	(4.74%)	
20	4639.68	4488.53	4536.47	4272.99	4031.69
	(15.08%)	(11.33%)	(12.52%)	(5.98%)	

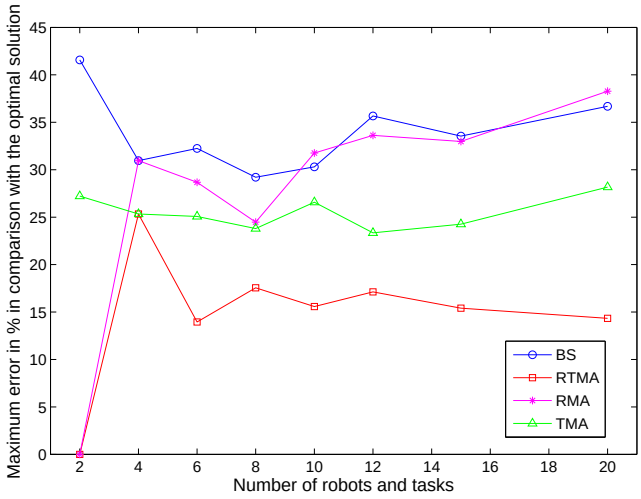


Figure 5: Mean of the maximum errors in percentage in comparison with the optimal solution in 100 simulations for the different types of algorithms and calculating the inital positions of the robots and the points of the formations at random.

optimal solution over all the simulations. In Figure 5, the maximal errors in percentage is shown. First of all, it can be observed that the RMA algorithm obtains worse maximal errors than the TMA algorithm and, in some cases, even worse than the BS algorithm, but the mean of the global cost is lower for the RMA algorithm as can be seen in Table 1. Therefore, the RMA algorithm has a better behavior on average but in certain circumstances the results can be worse than the TMA and BS algorithms. On the other hand, the BS algorithm is still the worst one for most of the cases, while the RTMA algorithm presents the best results. As can be seen in Figure 5, the mean of the maximum errors considering all the cases is 14.91% for the RTMA algorithm and 33.77% for the BS algorithm (which is greater than the mean error commented in Table 1). That means these algorithms do not have a constant behavior and for a specific situation, results could be worse than the average.

All the results presented have been calculated using random position of the robots and random points of the formations uniformly distributed. However, the quality of the solution for some of the algorithms depends on the type of formations. In Figure 6, there are two types of formations: the one on the left is calculated totally at random

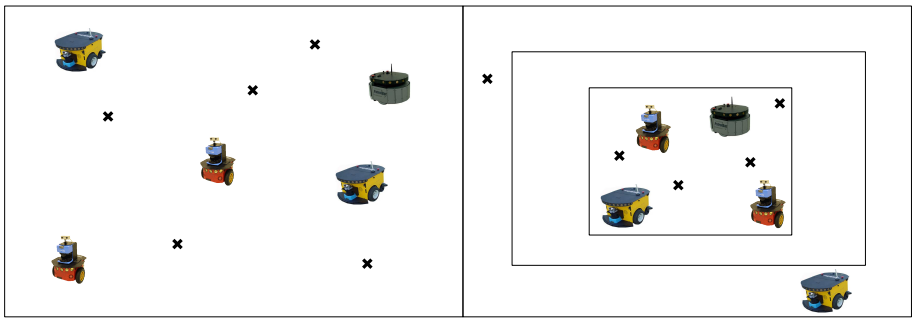


Figure 6: Types of formations used in the simulations. Left: initial positions of the robots and the formations calculated at random. Right: most of the formation points and the initial positions of the robots calculated at random in the small box and the others calculated outside the big box and calculated also at random.

and is the one used so far, the other formation on the right has a structure formed by two boxes. Most of the points and robots of the formation are in the small box, and the others outside the big box. As can be seen in Figure 7, the BS algorithm obtains worse results than the ones obtained with the other type of formation, specially for low number of robots and tasks. Another important characteristic of this type of formations is that the error in percentage in comparison with the optimal solution remains more or less constant for different number of robots and tasks. Therefore, for this type of formations the behavior of the algorithms for a specific situation are more predictable than with the totally random formations. Finally, the RTMA algorithm obtains also the best results while the BS algorithm the worst ones and, unlike the first type of formations, the TMA algorithm always obtains worse results than the RMA algorithm for all the cases simulated.

2.4.1 Different number of robots and tasks

In this section, our simulations always had the same number of robots and tasks. We are interested more in this case since our problem under study is the change of formation that involves all the robots. However, our algorithms can perfectly handle situations where the number of robots and tasks are different.

When there are more robots than tasks, the tasks will be allocated to the best

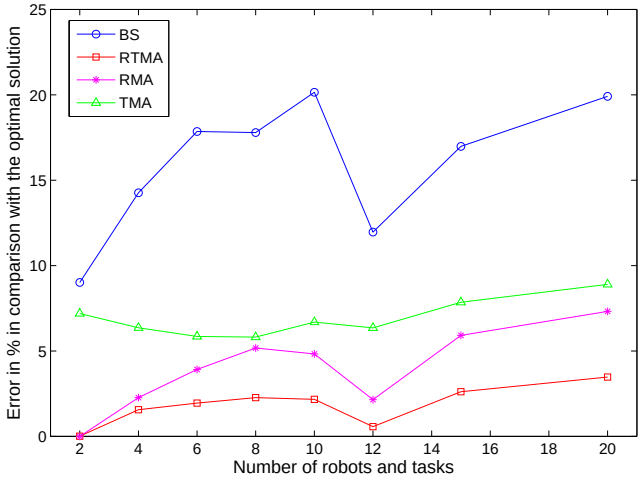


Figure 7: Mean error in percentage in comparison with the optimal solution for the different types of algorithms and calculating the inital positions of the robots and the points of the formations as it is described in the right part of the Figure 6 over 100 simulations.

robots and the rest of the robots will stay idle waiting for new tasks. In this case, the global cost will decrease since the probability to have robots close to the tasks increases, just for the simple reason that there are more robots.

On the other hand, when there are more tasks than robots, only the best tasks will be allocated to the group of robots. The remaining tasks will not be executed. Depending on the application these tasks will be lost, transmitted to a monitoring center where a human operator will decide what to do or saved to be allocated later. Also, in this case, the global cost will decrease since only the best tasks are executed by the different robots.

Simulations with different number of robots and tasks for the BS-WR and BS algorithms can be found in Section 3.5.

2.5 Experiments with real robots

2.5.1 Description of the testbed

We used a team of six mobile robots in order to implement our task allocation algorithms. Each of these robots is based on the iRobot Create platform (see Figure

Figure 8: Robot used in the experiments. iRobot Create with micro linux computer, wireless communication and GPS.

Figure 9: Team of robots running one of the experiments in an arena of 10x10m^2.

8) and we have added a micro linux computer (Connex 400XM processor from Gumstix), wireless capabilities and a GPS. Since the quality of the GPS alone is not good enough, we used a Kalman Filter [74] to combine the local odometry and the GPS measurements to obtain a decent global localization. Also, we have used a multi-robot architecture [52] that eases the integration of robots with high level algorithms such as our task allocation algorithms. Finally, we have tested our robots in a 10x10m^2 arena (see Figure 9).

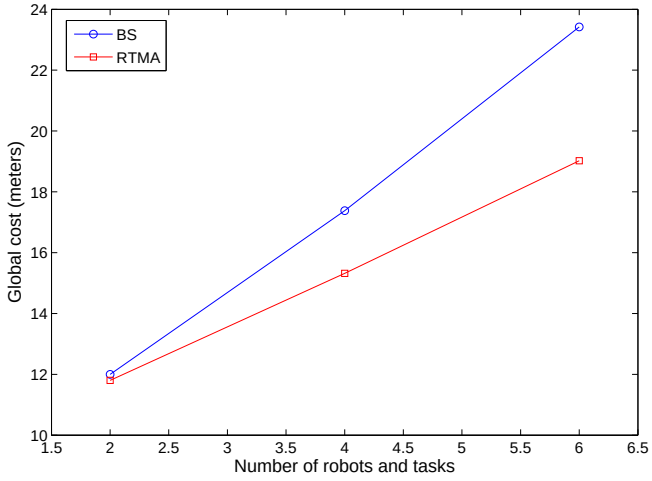


Figure 10: Results from the experiments in an arena of 10x10m^2 (mean of the global cost). The BS and RTMA algorithms have been tested with 2, 4 and 6 real robots, obtaining results similar to the simulated ones.

2.5.2 Results from experiments

Since experimentation is tedious and slow, only experiments with the BS and RTMA algorithms have been performed. We have chosen these two algorithms since the RTMA algorithm obtains the best results and the BS algorithm is a simple algorithm that can be used as a benchmark.

Experiments with different number of robots have been performed. Specifically, four experiments have been run with two robots, six with four robots and eight with six robots. In total, 18 experiments have been run with each of the two algorithms. All these experiments have been performed in an 10x10m^2 arena where the positions of the robots and tasks have been calculated at random.

In Figure 10, the results from the experiments are shown. It can be seen that these results follow the same dynamic as the simulation results where the RTMA algorithm obtains better results than the BS algorithm and the difference between both of them increases with respect to the number of robots and tasks.

2.6 Synchronization

Usually market-based approaches that can allocate more than one task to each robot use marginal costs [16, 65] as bids. In those cases, every robot has a local plan that describes the order of execution of the different tasks allocated to the robot. The cost of the local plan is the sum of the individual costs of all the tasks that composed it. Therefore, the cost of a task, its marginal cost, will be the difference between the cost of the local plan considering the task currently on auction and the cost of the plan itself. In these cases, better results are obtained if the auctions are not run concurrently [18] since it is crucial to know the current state of the local plan in order to calculate an updated marginal cost. For example, in Figure 11 a situation is shown where robots are participating in two auctions at the same time. Due to the fact that both auctions are run at the same time, the marginal costs are not correct since robots do not know yet if they will win any of these tasks and, therefore, are not considered in the calculation of the marginal cost. However, when the auctions are run sequentially, the marginal costs are always correct and the results obtained are better as can be seen in Figure 11.

In order to force the auctions to be run sequentially, a synchronization mechanism is needed. Usually a token-based algorithm is used where only the robot with the token can start an auction. However, the need for serialized auctions slows down the allocation algorithm and also increments the number of messages needed.

In the Initial Formation Problem, each robot has to execute only one task. Therefore, there is no need to have a local plan in each robot and also the marginal costs are not used. These characteristics allow the same quality of results to be obtained with and without synchronization. In order to illustrate this result we have implemented both the BS and RMA algorithms with and without synchronization. The synchronization has been implemented by means of a token protocol where only the robot with the token can start an auction. As it can be seen in Table 2, the results

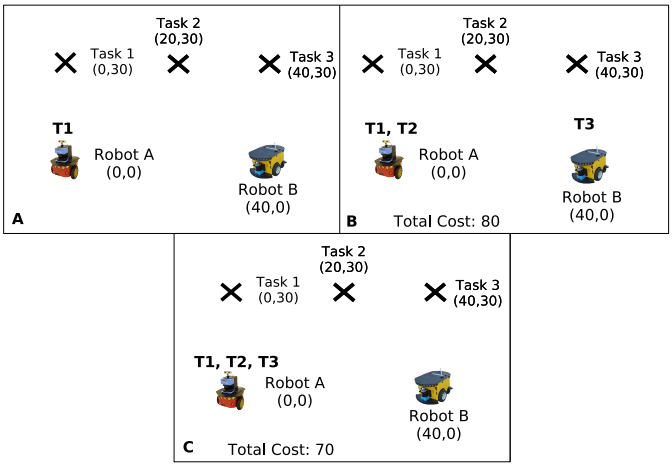


Figure 11: Figure A shows the initial situation where Robot A has already won Task 1. If tasks 2 and 3 are published at the same time and the auctions run concurrently, the marginal cost for Task 2 will be 20 for Robot A and 36.05 for Robot B. While the marginal costs for Task 3 will be 40 for Robot A and 30 for Robot B. Therefore, in Figure B it is shown the final allocation where tasks 1 and 2 are allocated to Robot A and task 3 to Robot B. In Figure C it is shown the final allocation when the auctions are run sequentially, obtaining a lower global cost than previously.

obtained for the BS and the RMA algorithms are almost the same with and without the token protocol. The maximum error for all the cases and both algorithms is 1.9%, which is negligible.

The possibility of using the algorithms for the Initial Formation Problem without any kind of synchronization mechanism makes these algorithms run much faster because the negotiations can be run in parallel. Also this fact reduces the number of messages used in the task allocation algorithm because it saves all the messages

Table 2: Results obtained with the BS and RMA protocol with and without synchronization over 100 simulations per each case.

Tasks & Robots	***BS***	***BS without synchronization***	***RMA***	***RMA without synchronization***
3	909.35	909.35	886.84	886.84
4	1473.52	1473.29	1436.16	1450.98
6	2020.13	2055.72	1958.49	1973.69
8	2443.57	2450.10	2365.30	2373.77
10	2865.81	2901.01	2711.63	2771.67
12	3233.25	3245.45	3144.98	3148.67

used in the token algorithm. In Figure 12, the difference, in execution time, is shown between the RMA run with and without the token protocol. It can be observed how the difference gets bigger with respect to the number of robots and tasks and with 12 robots the RMA algorithm without token is 32.34% faster than the algorithm with the token protocol. The two main sources of delays in a market-based algorithm with a token mechanism are:

- Wait time for bids (T_b): due to the fact that the number of robots is not known a priori, the auctioneer must wait a certain amount for the bids to be considered in the auction. When a synchronization method is used, the auctions are serialized, so the time to finish N auctions is at least $T_b \cdot N$. While in algorithms without synchronization mechanisms, the auctions are run in parallel. Due to this fact the difference between an algorithm with and without synchronization mechanism increases with respect to the number of robots. With more robots the number of auctions increases and, also, more number of auctions can be run in parallel.

- Time to request the token again (T_k): in our implementation, the robots that need to start an auction must ask for the token first. It could happen that the token cannot be passed because another robot is in the middle of an auction or has failures. Therefore, the robot waits for a certain amount of time, T_k, before it asks for the token again. It is important to point out that the main purpose of this timer is the reduction of the communication traffic. However, if the auction is finished just after someone asked for the token, the next auction will not start for at least T_k seconds. This delay can be reduced if the robot with the token remembers at least one of the robots that has asked for the token during the auction sequence and when the auction is finished, it can send the token immediately to that robot at the start of the next auction.

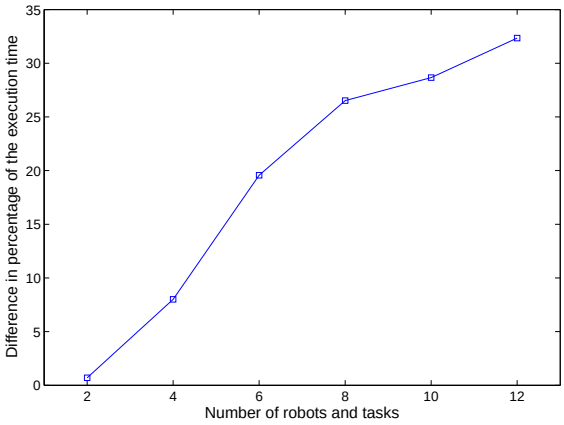


Figure 12: Difference in percentage of the execution time between algorithms with and without the synchronization mechanism ($T_b = 500ms$ and $T_k = 300ms$). The results are shown for different number of robots and tasks.

It is important to point out that algorithms without any synchronization method are more robust and fault tolerant because they are simpler. For example, if we are using a token-based synchronization method, there must be mechanisms to recover the token when it is lost.

2.7 Conclusions

This chapter presented five different algorithms that solve the Initial Formation Problem in a distributed way. The first two are a basic implementation of a market approach and it has been demonstrated the importance of reallocations since it improves the quality of the solution and makes the final allocation independent from the task announcements order.

Next, the BS algorithm has been compared with three improved algorithms. The BS algorithm is the simplest algorithm but obtains the worst results in most of the cases. Also, it is the algorithm that is most affected by the structure of the formation due to the fact that its results get worse with the second type of formations. The second and third algorithms use the mean of the costs (considering all the tasks associated to a robot or all the robots for a specific task) in order to increase the

information about the whole system and improve the results, but always keeping the distributed computation of the algorithm. These two algorithms obtain similar results for all the cases and better than the ones obtained with the first algorithm. Finally, the RTMA, which is a combination of the RMA and TMA algorithms, obtains the best results in all the cases for both types of formations since combines the good characteristics of the RMA and TMA algorithms.

Two different types of formations have been used. In the first one, the error in comparison with the optimal solution increases in a linear way with respect to the number of robots and tasks. In the sencod one, the error is kept slightly constant. Therefore, the behavior of the algorithms is more predictable for the second type of formations. Also, it has been proven that a synchronization mechanism is not needed for the Initial Formation Problem. Due to this fact our algorithms are much simpler and, also, they run faster.

Finally, the BS and RTMA algorithms have been validated in experiments with real robots obtaining results similar to the simulations.

CHAPTER III

PERFORMANCE EVALUATION USING PROBABILISTIC ANALYSIS

This chapter proposes a general probabilistic analysis approach for market-based algorithms that solve the Initial Formation Problem and can be used to compare different algorithms in different scenarios. The probabilistic analysis is used to calculate the expected value of the global cost, and is used as a metric to compare different algorithms. The probabilistic analysis is general and does not require any supposition, but the probabilistic distribution of the costs must be known.

The chapter is structured as follows. First, each task allocation algorithm is transformed into a centralized greedy algorithm that is applied to a matrix representation of the problem. Then, probabilistic analysis of the algorithms will be presented where the expected value of the global cost for any cost distribution is calculated. Next, the results of the analyses are applied to two different scenarios: random change of formation and dispersion. Moreover, the implications that have to be considered when the number of robots and tasks are not the same are exposed. Finally, our results are validated with simulations and real experiments.

3.1 Related work on performance evaluation

Although the efficiency of market-based algorithms has been evaluated in both simulation and some real implementations [19, 29], none of these works has obtained a theoretical bound on the real efficiency of these algorithms. As far as we know, the only work to obtain a bound for a market-based algorithm is [45] but they suppose that all the robots know all the desired positions from the beginning. Therefore, their

decentralized implementation differs from the classical market-based approach and computes the bids using the global information of the desired positions plus the local information of the robot. Also, a generic framework for auction-based multi-robot routing that studies theoretical guarantees for different bidding rules and different team objectives can be found in [46]. There are other recent works on theoretical bounds but the tasks also have to be communicated to all the robots at the beginning and they are not based on auctions. A distributed heuristic algorithm with local communication is explained in [83], while in [84] the agents are controlled by hybrid models using distributed potential fields. Both approaches fail to return a highly efficient solution since more than one robot can execute the same task. In [73], a solver of the Traveling Salesman Problem (TSP) is used to decide which robot should execute which task. However, this approach first solves a much more difficult problem to obtain a solution to the assignment problem.

All of the commented bound analyses focus on obtaining a worst case bound which is usually very pessimistic and may not ever happen in a real implementation scenario. The work presented in this chapter is unique in the sense that it calculates the expected value of the global cost that can be seen as a performance metric. The metric is more realistic than the worst case value, and provides an estimate of the performance over time.

3.2 Relation between Market-Based and Greedy Algorithms

Although the analysis could be applied to all the algorithms commented in Chapter 2, only its application to the BS-WR and BS algorithms is explained in detail. In this section, the equivalency between a distributed market-based algorithm and a centralized greedy algorithm will be described. This equivalency allows us to apply the probabilistic analysis to the greedy algorithm and extend the results to the distributed task allocation algorithm.

3.2.1 BS-WR: BaSic market-based algorithm Without Reallocations

This algorithm is equivalent to the column-scan method [44] for the assignment problem expressed as a matrix where each element is the cost associated with the respective robot and task. We consider that tasks are the columns of the cost matrix and robots the rows. In this algorithm, each column of the matrix is examined and the row with the lowest cost is selected. The selected row is marked and no longer examined for the rest of the algorithm. Through this process, the algorithm functions as follows:

1. Each column is scanned.
2. The smallest element of the column is selected.
3. The column and the row associated to this element are deleted and not considered for the rest of the algorithm.
4. The same procedure is repeated for the next column until all the columns have been scanned.
5. The selected elements are the solution of the problem and the global cost is the sum of these elements.

A simple example will be used to show how both algorithms obtain the same solution:

- The initial positions of the robots and the desired positions of the formation are the ones show in Figure 13.
- The matrix that models this specific problem is:

$$\begin{pmatrix} 30.0 & 41.23 & 20.0 \\ 50.0 & 10.0 & 44.72 \\ 80.0 & 72.11 & 30.0 \end{pmatrix}$$

- Following the algorithm steps, the smallest element of the first column is selected. This element is 30.0 which assigns robot A with task number 1. The row and column of the selected element is deleted and the following matrix is obtained:

$$\begin{pmatrix} 10.00 & 44.72 \\ 72.11 & 30.0 \end{pmatrix}$$

- Next the smallest element of the second column is selected. This element is 10.0 and therefore, the robot B is assigned to task number 2.

- Finally, the last assignment is made such that robot C is assigned to task number 3. The global cost for this problem will be $GC(3) = 70.0$.

As can be observed in Figure 13 and 14, the solution obtained with the basic market-based algorithm without reallocations is exactly the same as the one obtained by the column-scan method. Thus, the same bound can be applied to both algorithms. Also, it can be seen clearly how a change in the order of the tasks will produce a different allocation. For example, if Task 3 is announced before Task 1, the final allocation will be the same as Figure 15. This is not a desirable characteristic since it is difficult to predict the performance of the final allocation.

3.2.2 BS: BaSic market-based algorithm

As was stated in the previous section, the basic market-based algorithm uses reallocations to obtain the same solution irrespective of the order in which the tasks are announced. This algorithm is equivalent to the matrix-scan algorithm [44] that solves the assignment problem when it is expressed in a matrix form. The algorithm using this construct works as follows:

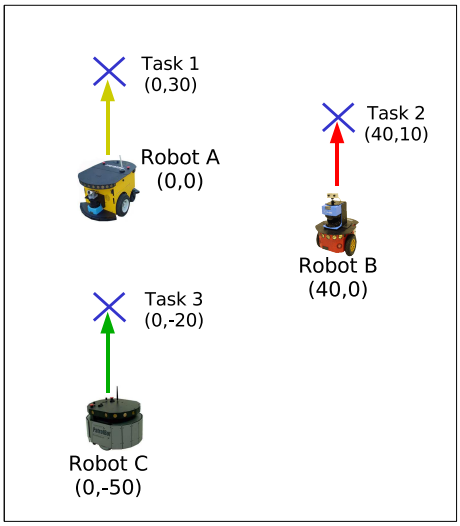


Figure 13: Initial position of the robots and the desired positions of the formation, and also, the final assignment obtained with the BS-WR algorithm.

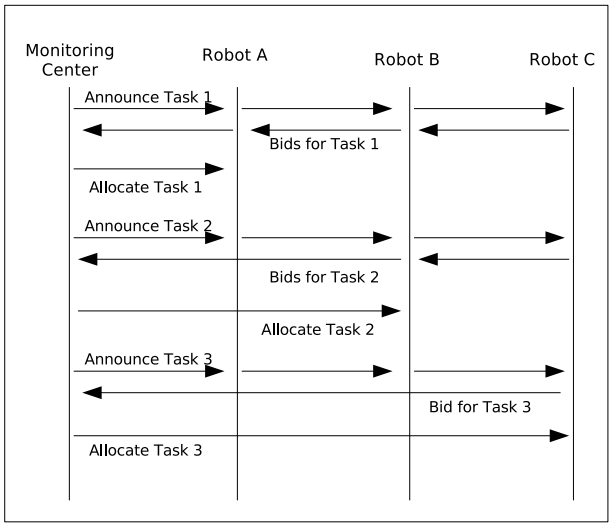


Figure 14: Messages exchanged in the auction process among the different robots using the BS-WR algorithm. The initial positions of the robots and the positions of the formations are the same as Figure 13.

1. The smallest element of the entire matrix is selected.

2. The row and column associated with this element are deleted and therefore, the order of the matrix is reduced by one.

3. The matrix is searched again for the smallest element and the process is repeated until a matrix of order one is reached.

4. The selected elements are the solution of the problem and the sum of them is the global cost.

The use of reallocations in the basic market-based algorithm ensures that it will obtain the same solutions as the matrix-scan algorithm. This fact is illustrated with the following example:

- The initial positions of the robots and the desired positions of the formation are the ones show in Figure 15.

- Supposing that the columns represent tasks and the rows robots, the matrix that models this specific problem is:

$$\begin{pmatrix} 30.0 & 41.23 & 20.0 \\ 50.0 & 10.0 & 44.72 \\ 80.0 & 72.11 & 30.0 \end{pmatrix}$$

- Following the algorithm steps, the smallest element of the matrix is selected. This element is 10.0 which assigns robot B with task number 2. The row and column of the selected element is deleted and the following matrix is obtained:

$$\begin{pmatrix} 30.0 & 20.0 \\ 80.0 & 30.0 \end{pmatrix}$$

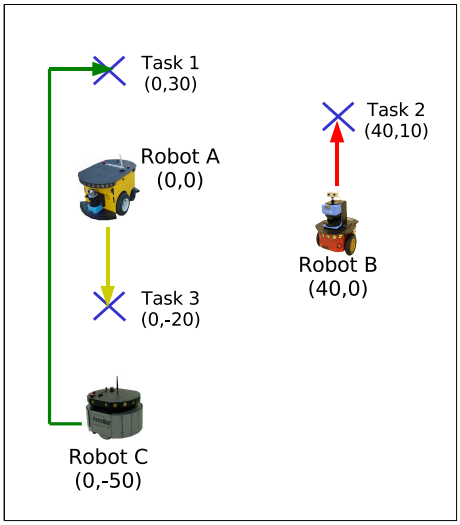


Figure 15: Initial position of the robots and the desired positions of the formation, and also, the final assignment obtained with the BS algorithm.

- Again the smallest element of the new matrix is selected. This element is 20.0 and, therefore, robot A is assigned to task number 3.

- Finally, the last assignment is made such that robot C is assigned to task number 1. The global cost is $GC(3) = 110.0$.

Figure 16 illustrates the increase in messages exchanged among the robots running the basic market-based algorithm for the problem stated in Figure 15. As can be observed, the solution obtained with the basic market-based algorithm is exactly the same as the one obtained by the matrix-scan method. Thus, the same bound can be applied to both algorithms.

3.3 Probabilistic Analysis of the Market-Based Algorithms

The analysis of the algorithms will be based on a probabilistic approach. The key idea is to calculate the expected value of the global cost that provides an estimate of the performance of the algorithm over time. The probabilistic analysis approach is applied to the greedy algorithms but is valid also for our distributed market-based algorithms based on solution equivalency (as discussed in Section 3.2). In our probabilistic

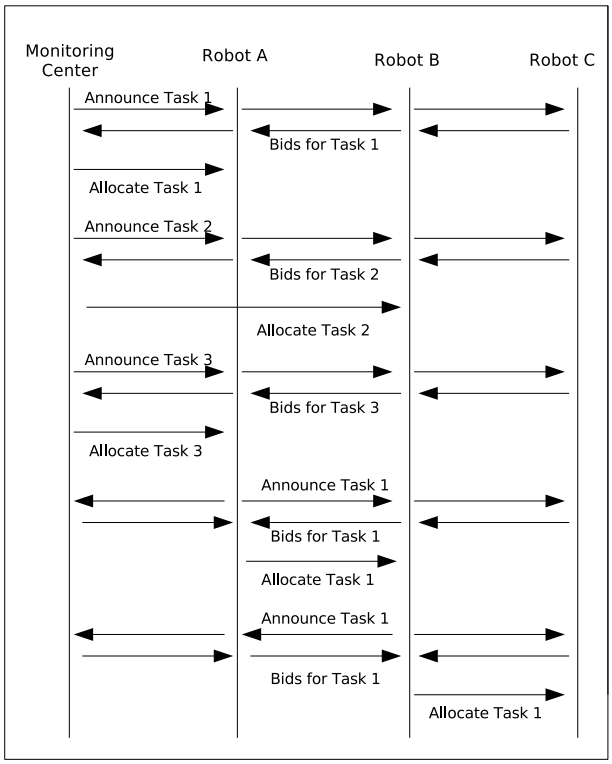


Figure 16: Messages exchanged in the auction process among the different robots using the BS algorithm. The initial positions of the robots and the positions of the formations are the same as Figure 15.

analysis approach, the Initial Formation Problem is formulated as a matrix where each cell represents the cost associated with a specific robot and task. These costs are modeled as random variables equally distributed and independent.

3.3.1 BS-WR: BaSic market-based algorithm Without Reallocations

The global cost for the BS-WR algorithm (see Section 3.2.1) is defined as $\sum_{k=1}^{n} m_k$ where m_k is the minimum element of the k^{th} column which has $n-k+1$ elements from the cost matrix and n is the size of the cost matrix as well as the number of robots and tasks. We define M_k as the minimum of $n-k+1$ independent and equally distributed random variables ($X_{i,k}$) of the k^{th} column, i.e.,

$$M_k \equiv \min\{X_{1,k}, X_{2,k}, X_{3,k}, \ldots, X_{(n-k+1),k}\}.$$

Its cumulative distribution function is given by

$$F_{M_k}(x) = 1 - [1 - F_X(x)]^{n-k+1}$$

since all the random variables are equally distributed and independent. $F_X(x)$ is the cumulative distribution function of each of the random variables, $X_{i,k}$. The resulting probability density function of M_k is

$$f_{M_k}(x) = (n-k+1)\,[1 - F_X(x)]^{n-k}\, f_X(x)$$

where $f_X(x)$ is the probability density function for each of the random variables, $X_{i,k}$, $i \in [1, n-k+1]$.

Once we know the probability density function, the expected value of M_k can be calculated as

$$E(M_k) = \int_{-\infty}^{\infty} x \cdot (n-k+1)\,[1 - F_X(x)]^{n-k}\, f_X(x)\, dx. \tag{5}$$

Finally, the expected value of the global cost is defined as the expected value of the sum of the n random variables $\{M_1, M_2 \ldots, M_n\}$, since the expected value is a linear

operator

$$E_{GC}(n) = E(M_1 + M_2 + \dots M_n) = \sum_{k=1}^{n} E(M_k).$$

Then,

$$E_{GC}(n) = \sum_{k=1}^{n} \int_{-\infty}^{\infty} x \cdot (n-k+1) \left[1 - F_X(x)\right]^{n-k} f_X(x)\, dx \tag{6}$$

where, as we commented before, $F_X(x)$ is the cumulative distribution function and $f_X(x)$ is the probability density function of any of the n^2 random variables that form the cost matrix that models our Initial Formation Problem. Therefore, in order to calculate the expected value of the global cost for the BS-WR algorithm we just need to know which distribution the costs follow.

3.3.2 BS: BaSic market-based algorithm

From Section 3.2.2 we know that in order to compute the global cost we have to calculate the minimum value of the matrix of order n (number of robots and tasks). Then, we remove the row and the column of the minimum and obtain a matrix of order $n-1$. We continue this process until the matrix is empty. The global cost is the sum of the calculated minimums. As was stated before, the cost matrix is formed initially with n^2 independent and equally distributed random variables, $\{X_1, X_2, \dots, X_{n^2}\}$. In order to simplify the explanation, we have changed the notation of the random variables such that they are specified as a vector of size n^2. We define $M_1 \equiv \min\{X_1, X_2, \dots, X_{n^2}\}$. Then, the cumulative distribution function is given by

$$F_{M_1}(x) = 1 - \left[1 - F_X(x)\right]^{n^2}.$$

The expected value, μ_1, is then determined as

$$\mu_1 = E(M_1) = \int_{-\infty}^{\infty} x \cdot n^2 \left[1 - F_X(x)\right]^{n^2-1} f_X(x)\, dx \tag{7}$$

where $F_X(x)$ and $f_X(x)$ are the cumulative and density functions of each of the random variables X_i, $i \in [1, n^2]$. Next, we remove $2n-1$ variables (same row and

column of the minimum), and afterwards, we calculate again the minimum value of the new matrix with order $n-1$. This will lead to a new variable M_2 defined as the minimum of the left variables, $M_2 \equiv \min\{X_1, X_2, \ldots, X_{(n-1)^2}\}$. Notice that M_2 depends on the first minimum selected M_1 and therefore, the expected value of M_2 can be computed as

$$\mu_2 = E(M_2) = E(E(M_2|M_1)).$$

We define $h_{M_2}(M_1) = E(M_2|M_1)$. It is important to take into account that $h_{M_2}(M_1)$ is a random variable depending on M_1 and not a real number, and thus, $E(M_2)$ will be a function depending on $E(M_1)$. The expected value of the global cost is defined as the expected value of the sum of $\{M_1, M_2 \ldots, M_n\}$, so we repeat the previous procedure to obtain

$$E_{GC}(n) = E(M_1 + M_2 + \ldots M_n) = \sum_{k=1}^{n} E(M_k) = \mu_1 + \sum_{k=2}^{n} E(h_{M_k}(M_{k-1})). \tag{8}$$

Formula (8) simplifies notably if $h_{M_k}(\cdot)$ is a linear function of the form $h_{M_k}(x) = a_k x + b_k$, for $k \in [2, n]$. Then,

$$\mu_k = E(M_k) = E(h_{M_k}(M_{k-1})) = h_{M_k}(E(M_{k-1})) = h_{M_k}(\mu_{k-1}).$$

And (8) becomes

$$E_{GC}(n) = \mu_1 + \sum_{k=2}^{n} h_{M_k}(\mu_{k-1}).$$

Notice that μ_k can be expressed in a recursive way as a function of μ_1

$$\begin{aligned} \mu_k = h_{M_k}(\mu_{k-1}) &= a_k\mu_{k-1} + b_k = a_k\left(a_{k-1}\mu_{k-2} + b_{k-2}\right) + b_k = \ldots = \\ &\mu_1 \prod_{i=2}^{k} a_i + \sum_{i=2}^{k} \left(b_i \cdot \prod_{j=i+1}^{k} a_j \right). \end{aligned}$$

Finally, the expected value of the global cost can be expressed as

$$E_{GC}(n) = \mu_1 + \sum_{k=2}^{n} \left[\mu_1 \prod_{i=2}^{k} a_i + \sum_{i=2}^{k} \left(b_i \cdot \prod_{j=i+1}^{k} a_j \right) \right]. \tag{9}$$

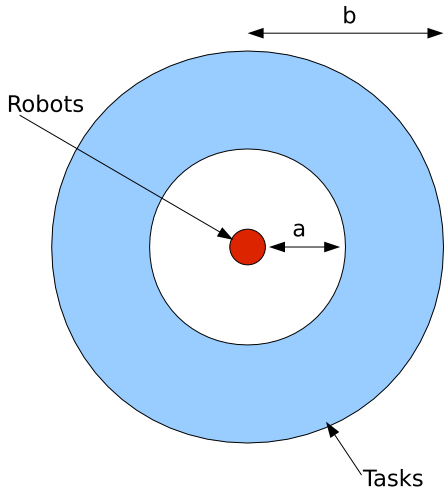


Figure 17: Dispersion scenario with costs uniformly distributed between $[a, b]$. Robots are within the red circle and the tasks are distributed at random within the blue doughnout.

3.4 Application of the Analysis to Different Scenarios

For the following section we focus on an exploration application in different types of scenarios. The tasks are waypoint tasks and the costs are defined as the euclidean distance in 2D between the robot and the point to which the robot is expected to navigate. The results obtained in the previous section are completely general and can be used with other definitions of costs such as the consumed energy or the euclidean distance considered in three dimensions.

3.4.1 Dispersion Scenario

The dispersion scenario describes a situation when we have a team of robots that are deployed together and want them to disperse around an area. For example, imagine that we send a team of robots to Mars, and after the landing, we want them to disperse so they can explore the area. In this scenario, the costs follow a uniform distribution between $[a, b]$. In this case the positions of the new formation are at least a distance a from the original position of the robots (see Figure 17).

3.4.1.1 Expected Value for the BS-WR Algorithm

Let us suppose that the costs are uniformly distributed between $[a, b]$, i.e., $X_{i,j} \sim U(a, b)$. Applying Formula (6), the expected value of the global cost is

$$E_{GC}(n) = a \cdot n + \sum_{k=1}^{n} \frac{b-a}{(n-k+1)+1} = a \cdot n + \sum_{k=1}^{n} \frac{b-a}{k+1}. \tag{10}$$

It can be seen that the term $a \cdot n$ is dominant in the expected value of the global cost. This means that when the number of robots and tasks increases, $E_{GC}(n)$ is going to increase linearly with n. Furthermore, the larger the number a, the bigger the slope will be of the linear dependency. One special case can be studied when $a = 0$, the expected value of the global cost is now

$$E_{GC}(n) = b \sum_{k=1}^{n} \frac{1}{k+1} = b \cdot (H_{n+1} - 1) \simeq b \cdot \left(\ln(n+1) + \gamma + \frac{1}{2 \cdot (n+1)} - 1 \right)$$

where H_{n+1} is the harmonic number of the first $n+1$ natural numbers and γ [1] is the Euler-Mascheroni constant. In this case, $E_{GC}(n)$ increases with the logarithm of the number of robots or tasks.

3.4.1.2 Expected Value for the BS Algorithm

Let us suppose that the costs are uniformly distributed between $[a, b]$, i.e., $X_i \sim U(a, b)$. Then,

$$F_X(x) = \frac{x-a}{b-a}, \quad \text{for } a \leq x < b$$

which yields to

$$F_{M_1}(x) = 1 - \left[1 - \frac{x-a}{b-a} \right]^{n^2}, \quad \text{for } a \leq x < b.$$

We are firstly interested in computing $E(M_2|M_1)$ so we compute the cumulative distribution function

$$P(M_2 < x|m_1) = 1 - \left[\prod_{i=1}^{(n-1)^2} P(X_i > x|m_1) \right].$$

[1] $\gamma = \lim_{n\to\infty} \left[\sum_{k=1}^{n} \frac{1}{k} - \ln n \right] \simeq 0.577$

It can be easily seen that

$$X_i|M_1 \sim U(M_1, b), \quad \text{for } i = 1, \ldots, n^2$$

and thus,

$$P(M_2 < x|m_1) = 1 - \left[1 - \frac{x - m_1}{b - m_1}\right]^{(n-1)^2} \quad \text{for } m_1 \leq x < b.$$

By (11) and the previously computed density function, it follows that

$$h_{M_2}(M_1) = E(M_2|M_1) = \frac{(n-1)^2}{(n-1)^2 + 1} M_1 + \frac{b}{(n-1)^2 + 1}$$

which implies

$$\mu_2 = E(M_2) = \frac{(n-1)^2}{(n-1)^2 + 1} \mu_1 + \frac{b}{(n-1)^2 + 1}.$$

In order to obtain the successive values μ_k we proceed similarly to obtain,

$$h_{M_k}(M_{k-1}) = E(M_k|M_{k-1}) = \frac{(n-k+1)^2}{(n-k+1)^2 + 1} M_{k-1} + \frac{b}{(n-k+1)^2 + 1}.$$

Since $h_{M_k}(\cdot)$ is linear, the expected value of the global cost can be calculated using Formula (9). In order to apply this formula, we also need to calculate μ_1 using (7)

$$\mu_1 = \frac{n^2 a + b}{n^2 + 1}.$$

Thus,

$$E_{GC}(n) = \frac{n^2 a + b}{n^2 + 1} + \sum_{k=2}^{n} \frac{n^2 a + b}{n^2 + 1} \prod_{i=2}^{k} \frac{(n-i+1)^2}{(n-i+1)^2 + 1} \quad + \\ \sum_{i=2}^{k} \frac{b}{(n-i+1)^2 + 1} \cdot \prod_{j=i+1}^{k} \frac{(n-j+1)^2}{(n-j+1)^2 + 1}. \tag{11}$$

3.4.2 Random Formation Scenario

In this scenario, the robots and tasks are initially positioned randomly in a square area. Usually, this is the scenario used to test task allocation algorithms related to exploration or navigation. However, in this scenario the costs do not follow a known distribution and, as far as we know, this is the first effort to model the distribution of the costs in this useful scenario.

3.4.2.1 Distribution of the euclidean distance for uniformly distributed points

Supposing we have calculated the positions of the robots and the points of the new formation at random, the distribution of the distances between each robot and point of the new formation is required for analysis.

So the position of the robots (X_r, Y_r) are random variables that follow a uniform distribution. For simplification and without losing generality, let X_r and Y_r be in the range of $[0, 1]$. The position of the tasks (X_t, Y_t) also follow uniform random variables between $[0, 1]$. We suppose that all the variables are independent. The distribution of $X \sim |X_r - X_t|$ is defined by the following cumulative distribution function

$$F_X(x) = 2x - x^2 \quad \text{if } 0 \leq x \leq 1. \tag{12}$$

Next, we are interested in the distribution of the random variable defined as

$$C \equiv \sqrt{X^2 + Y^2}$$

where X and Y are random variables defined by (12) and C is the random variable of the costs defined as a euclidean distance.

The cumulative distribution function of the new random variable, C, can be calculated as follows

$$F_C(c) = \int_{-\infty}^{\infty} F_X(\sqrt{c^2 - y^2}) f_Y(y) dy$$

which has to be defined by parts

$$F_C(c) = \begin{cases} 0 & \text{if } c \leq 0 \\ \displaystyle\int_0^c F_X(\sqrt{c^2 - y^2}) f_Y(y) dy & \text{if } 0 \leq c \leq 1 \\ \displaystyle\int_0^{\sqrt{c^2-1}} F_X(\sqrt{c^2 - y^2}) f_Y(y) dy + & \\ \displaystyle\int_{\sqrt{c^2-1}}^1 F_X(\sqrt{c^2 - y^2}) f_Y(y) dy & \text{if } 1 \leq c \leq \sqrt{2} \\ 1 & \text{if } c \geq \sqrt{2} \end{cases}$$

Solving these integrals, we obtain

$$F_C(c) = \begin{cases} 0 & \text{if } c \leq 0 \\ \frac{1}{2}c^4 - \frac{8}{3}c^3 + \pi c^2 & \text{if } 0 \leq c \leq 1 \\ 2c^2 \arcsin \frac{1}{c} - 2c^2 \arcsin \frac{\sqrt{c^2-1}}{c} + & \\ 2c^2\sqrt{c^2-1} - c^4 + \frac{2}{3}(c^2-1)^{\frac{3}{2}} + & \\ \frac{(c^2-1)^2}{2} - \frac{7}{6} + 2\sqrt{c^2-1} - (c^2-1) & \text{if } 1 \leq c \leq \sqrt{2} \\ 1 & \text{if } c \geq \sqrt{2} \end{cases} \tag{13}$$

Once we get the cumulative distribution function, the random variable that models the cost as an euclidean distance is totally defined and the probability density function can be calculated easily. However, it can be observed that the density function is not continuous for $c = 1$ and this will imply changes in the way to calculate the expected value.

3.4.2.2 Expected Value for the BS-WR Algorithm

Since the cost density function (13) is not continuous, we will use an alternative way of calculating the expected value. By definition

$$E(X) = \int x \cdot f_X(x)\, dx.$$

Using the integration by parts rule

$$E(X) = [x \cdot F_X(x)] - \int F_X(x)\, dx. \tag{14}$$

Therefore, applying the above rule in Formula (6) and with the change of variable $k = n - k + 1$, the expected value of the global cost is

$$E_{GC}(n) = \sum_{k=1}^{n} A - \int_0^{\sqrt{2}} 1 - [1 - F_X(x)]^k \; dx \tag{15}$$

where $F_X(x)$ is the cumulative distribution function defined in (13) and

$$A = \left[x \cdot \left(1 - [1 - F_X(x)]^k\right)\right]_0^{\sqrt{2}}$$

which is a constant value.

As it will be explained in Section 3.6, this integral cannot be solved analytically and numerical methods have to be used.

3.4.2.3 Expected Value for the BS Algorithm

Following the steps explained in Section 3.3.2 and taking into account the results of Section 3.4.1.2, we calculate μ_1 using Formula (7) and the rule of integration by parts defined in (14) as

$$\mu_1 = B - \int_0^{\sqrt{2}} 1 - [1 - F_X(x)]^{n^2} \, dx$$

where $F_X(x)$ is defined in (13) and B is a constant value equal to

$$B = \left[x \cdot \left(1 - [1 - F_X(x)]^{n^2} \right) \right]_0^{\sqrt{2}}.$$

Next, we calculate $E(M_k|M_{k-1})$ which we have defined as

$$h_{M_k}(M_{k-1}) = C - \int_{m_{k-1}}^{\sqrt{2}} 1 - \left[\frac{1 - F_X(x)}{1 - F_X(m_{k-1})} \right]^{(n-k+1)^2} \, dx$$

where C is

$$C = \left[x \cdot \left(1 - \left[\frac{1 - F_X(x)}{1 - F_X(m_{k-1})} \right]^{(n-k+1)^2} \right) \right]_0^{\sqrt{2}}.$$

Finally, we compute the expected value of the global cost using (8).

3.5 Extension to different number of robots and tasks

All the results in the previous section have assumed the same number of robots and tasks. We are going to generalize our results for situations when the number of robots (n_R) and tasks (n_T) are different. The dispersion scenario will be used to illustrate this generalization, but it could be applied to any other scenario following an analogous mechanism.

3.5.1 More robots than tasks

In this case the cost matrix will not be squared and it will have more rows than columns. When the allocation is finished, there will be some robots that are idle. These robots will be the ones that have the highest costs for the group of tasks.

The column scan method, which is equivalent to the BS-WR algorithm, now has to scan n_T columns. The k^{th} column has $n_R - k + 1$ elements and there will be $n_R - n_T$ rows or robots at the end of the algorithm without a task allocated. Therefore, Formula (10) changes to

$$E_{GC}(n_T, n_R) = a \cdot n_T + \sum_{k=1}^{n_T} \frac{b-a}{(n_R - k + 1) + 1}. \tag{16}$$

Similarly, the matrix scan method (equivalent to the BS algorithm) stops after n_T steps, when all the tasks have been allocated but there are $n_R - n_T$ rows or robots without a task. For this case, M_k is the minimum of $(n_T - k + 1)(n_R - k + 1)$ random variables. Therefore, Formula (9) changes to

$$E_{GC}(n_T, n_R) = \mu_1 + \sum_{k=2}^{n_T} \left[\mu_1 \prod_{i=2}^{k} a_i + \sum_{i=2}^{k} \left(b_i \cdot \prod_{j=i+1}^{k} a_j \right) \right] \tag{17}$$

where μ_1,a_k and b_k are now

$$\begin{aligned} \mu_1 &= \frac{n_R n_T \cdot a + b}{n_R n_T + 1}, \\ a_k &= \frac{(n_R - k + 1)(n_T - k + 1)}{(n_R - k + 1)(n_T - k + 1) + 1}, \\ b_k &= \frac{b}{(n_R - k + 1)(n_T - k + 1) + 1}. \end{aligned} \tag{18}$$

3.5.2 Less robots than tasks

In this case the number of columns in the cost matrix is higher than the number of rows. There will be some tasks that will not be executed. These tasks will be the ones that have the highest costs for the group of robots.

The column scan method only scans the first n_R columns and the k^{th} column will have $n_R - k + 1$ elements. At the end of the algorithm, the last $n_T - n_R$ columns or

tasks will not be scanned or allocated to any robot respectively. Therefore, Formula (10) changes to

$$E_{GC}(n_R) = a \cdot n_R + \sum_{k=1}^{n_R} \frac{b-a}{n_R - k + 1}. \tag{19}$$

It can be observed that E_{GC} does not depend on n_T, since we just allocate the tasks in order. When there are no more robots left, we just discard the rest of the tasks. So, the problem is equivalent to having n_R number of robots and tasks.

However, all the tasks will be considered with the BS algorithm, since it uses reallocation. This fact increases the difference between both algorithms. The BS algorithm stops after n_R steps with $n_T - n_R$ tasks without being allocated. M_k will again represent the minimum of $(n_R - k + 1)(n_T - k + 1)$ random variables. Therefore, Formula (9) changes to

$$E_{GC}(n_R) = \mu_1 + \sum_{k=2}^{n_R} \left[\mu_1 \prod_{i=2}^{k} a_i + \sum_{i=2}^{k} \left(b_i \cdot \prod_{j=i+1}^{k} a_j \right) \right] \tag{20}$$

where μ_1,a_k and b_k take the same expressions as in (18).

3.6 Validation of the results with simulations

3.6.1 Description of the simulation environment

For testing purposes we have used the Matlab scripting language to implement the greedy algorithms that are equivalent to our task allocation algorithms. Also, we have implemented a multi-robot simulator that has been used to test our distributed algorithms. This simulator is based on an architecture designed for heterogeneous robots [78] and divided into three layers. The highest layer is independent from the type of robot and is the only one aware of the existence of other robots. Thus, the task allocation algorithm is implemented in this layer and can be used, without modification, in both simulations and real robots. Moreover, the inter-robot communication system implemented in this layer is based on IP, so it can also be used as an interprocess communication method for simulations. The other two layers are used

to execute the different tasks allocated to the individual robots making the creation of new algorithms easier by using a modular, component-based architecture. Finally, the multi-robot simulations have been used to valid the results of the matrix-based greedy algorithms [77].

3.6.2 Results from simulations

First, we validate the results obtained for the dispersion scenario. The expected value of the global cost (E_{GC}) for the BS-WR algorithm can be calculated using (10) and for the BS algorithm using (11). In Figure 18 the results when the costs follow a uniform distribution between $[5, 100]$ are shown and can be observed that our theoretical E_{GC} remains very close to the values obtained in simulation. Also, BS algorithm obtains better results than BS-WR algorithm proving that reallocation of tasks reduces the global cost. As was commented in Section 3.4.1.1, when the minimum value of the costs (a) is not zero, E_{GC} increases linearly with respect to the number of robots since all the robots, no matter which task that they choose, have to navigate at least a distance equal to a. The slope of this linear dependency is proportional to the value of a as can be seen in Figure 19.

A special case appears when $a = 0$. As can be observed from Figure 20, E_{GC} for both algorithms increases with the logarithm of the number of robots and tasks and our theoretical results also show this behavior. This case was explained for the BS-WR algorithm in Section 3.4.1.1.

Also our results model accurately the situations when the number of robots and tasks are not the same. When the number of robots is larger than the number of tasks, the tasks are allocated to the robots with the lowest costs and the rest of the robots will be idle, waiting for more tasks. Using Formulae (16) and (17) for BS-WR and BS algorithms respectively and having twice as many robots as tasks, Figure 21 shows that the results from our probabilistic analysis still remain close to

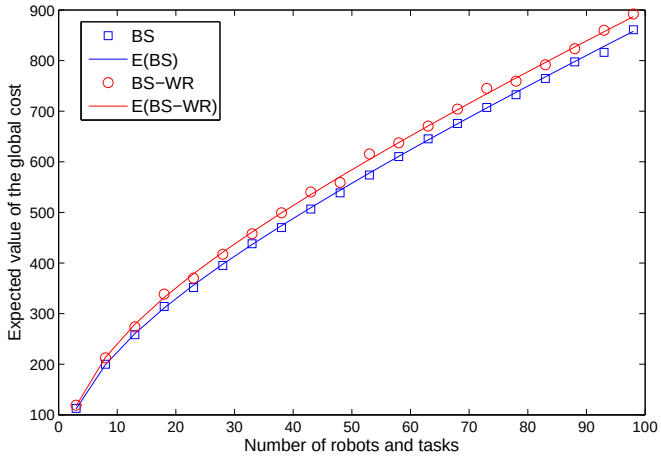


Figure 18: Expected value of the global cost over 100 simulations where the costs follow a uniform distribution between $[5, 100]$. The circles represent the results from simulation for the BS-WR algorithm and the squares for the BS algorithm. The theoretical results, E(BS) and E(BS-WR) respectively, are shown as solid lines.

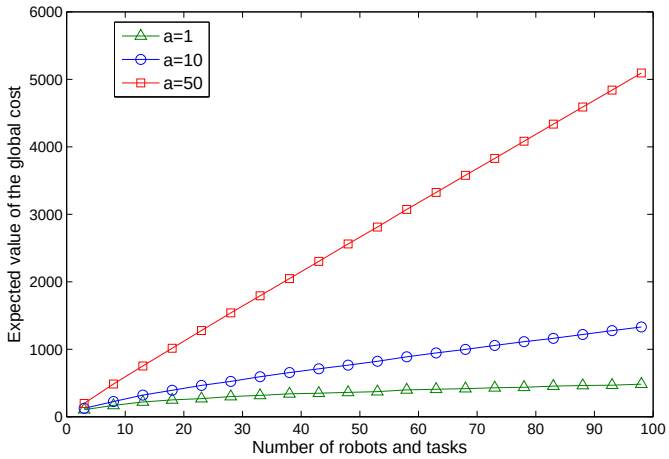


Figure 19: Expected value of the global cost over 100 simulations where the costs follow a uniform distribution between $[a, 100]$, being a equal to 1, 10 and 50. The slope of E_{GC} is directly proportional to a.

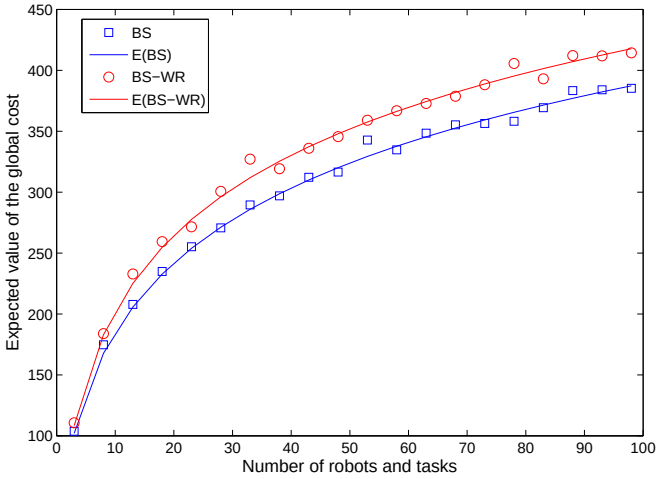


Figure 20: Expected value of the global cost over 100 simulations where the costs follow a uniform distribution between $[0, 100]$. In this case E_{GC} for both algorithms increases with the logarithm of the number of robots and tasks. The circles represent the results from simulation for the BS-WR algorithm and the squares for the BS algorithm. The theoretical results, E(BS) and E(BS-WR) respectively, are shown as solid lines.

the simulation results. The expected value of the global cost is smaller than in Figure 20 since only the robots with the lowest costs win a task. For the same reason, the difference between BS-WR and BS algorithms decreases. Another interesting fact is that the expected value seems to remain constant when the number of robots is large enough instead of increasing with the logarithm of n_R as happened when the number of robots and tasks are equal. From (16), when $n_R = J \cdot n_T$ for $J > 1$, it can be inferred that

$$\begin{aligned} E_{GC}(n_T) &= a \cdot n_T + (b-a)\left(H_{Jn_T+1} - H_{(J-1)n_T+1}\right) \\ &\overset{=}{_{n_T \to \infty}} a \cdot n_T + (b-a) \ln\left(\frac{J}{J-1}\right) \end{aligned}$$

where H_n is the harmonic number of the first n natural numbers. For the values used in Figure 21 ($a = 0$, $b = 100$ and $J = 2$), E_{GC} tends to 69.31 which matches with our simulated results. It can be seen here the potential of having an analytical formula of the performance of the algorithms. Other interesting facts, such as the

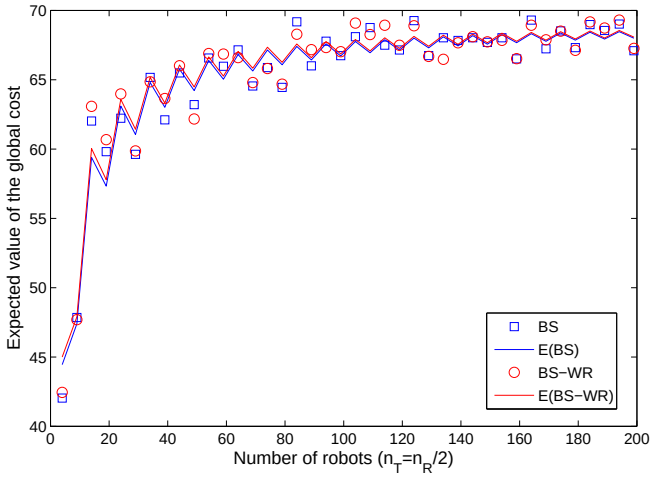


Figure 21: Expected value of the global cost over 100 simulations where the costs follow a uniform distribution between $[0, 100]$. The number of tasks is half the number of robots. The circles represent the results from simulation for the BS-WR algorithm and the squares for the BS algorithm. The theoretical results, E(BS) and E(BS-WR) respectively, are shown as solid lines.

relation between J and how fast the global cost tends to the constant value could be calculated, but for space reasons will be considered for future work.

When the number of robots is smaller than the number of tasks, only the tasks with lowest costs will be allocated to robots. The rest of the tasks will not be allocated to any robot. Using Formulae (19) and (20) for the BS-WR and the BS algorithms respectively, with twice as many tasks as robots, Figure 22 shows that the BS-WR algorithm performs much worse than the BS algorithm. Since the BS-WR algorithm does not use reallocation, only the first n_R tasks will be considered and the problem is equivalent to the one with n_R number of robots and tasks. However, using reallocations, BS algorithm takes into account all the tasks but only allocates the best n_R tasks to robots. That is the reason why E_{GC} for BS-WR algorithm increases with the logarithm of the number of robots and E_{GC} tends to a constant value as in the previous case.

Since we are working with the expected value or mean of the global cost, our results are close to the real costs when the number of experiments is large enough.

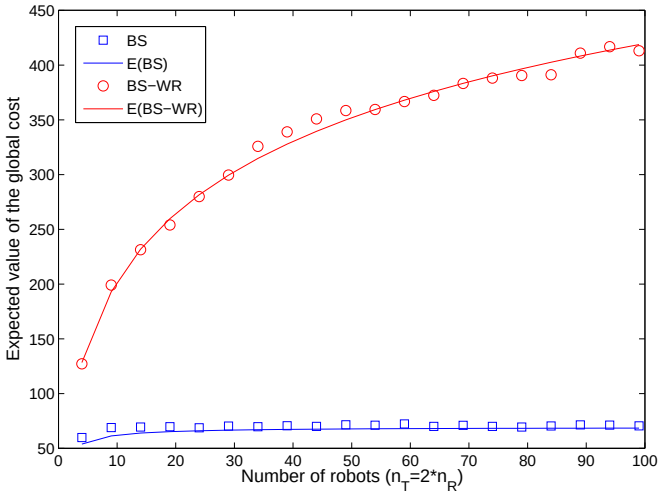


Figure 22: Expected value of the global cost over 100 simulations where the costs follow a uniform distribution between $[0, 100]$. The number of tasks is twice the number of robots. The circles represent the results from simulation for the BS-WR algorithm and the squares for the BS algorithm. The theoretical results, E(BS) and E(BS-WR) respectively, are shown as solid lines.

As can be seen from Figure 23, the difference between E_{GC} calculated theoretically and the one obtained from the simulations decreases with respect to the number of experiments. We would like to obtain a formula that will tell us how large the difference between our theoretical and experimental results could be depending on the number of experiments.

First, we need to model the sample mean of the global cost. Using the Central Limit Theorem (see Theorem 1), if the number of samples (in our case experiments) is sufficiently large, the sample mean follows a normal distribution and it can be bounded up to a percentage. Usually, it is considered that a sufficiently large number of samples is above 30. So, if we have run more than 30 experiments, it can be said that $(1-\alpha)\cdot 100\%$ of the sample means will lie between $t_{\alpha/2}/\sqrt{N}$ sample standard deviations of the population mean. Thus, for α values small enough, we can obtain an approximation of the maximal difference between E_{GC} calculated from our probabilistic analysis

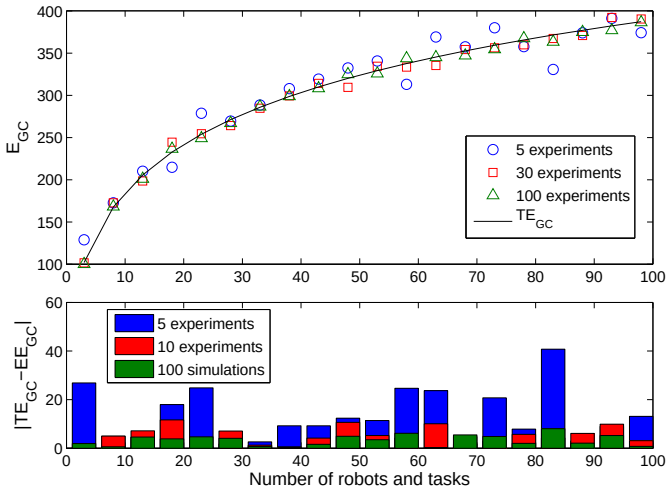


Figure 23: Expected value of the global cost for different number of missions. At the top, it can be seen the different expected values obtained from different number of missions. The solid line represents the theoretical value. At the bottom, the bars represents the difference between the expected value of the global cost from the simulations (sample mean) and from the theoretical result (population mean).

(TE_{GC}) and the one obtained from the experimental results (EE_{GC}) as

$$max(|TE_{GC} - EE_{GC}|) \gtrsim t_{\alpha/2}\frac{s}{\sqrt{N}} \tag{21}$$

where $t_{\alpha/2}$ is the t-value with an area $\alpha/2$ to its right (usually obtained from a table), N is the number of experiments and s is the sample standard deviation. We consider that $\alpha = 0.1$ is small enough since it means that 90% of the differences between TE_{GC} and EE_{GC} will be smaller than the right part of Formula (21). For $\alpha = 0.1$ and $N \geq 30$, we obtain

$$max(|TE_{GC} - EE_{GC}|) \gtrsim 1.282\frac{s}{\sqrt{N}}.$$

Theorem 1 (Central Limit Theorem) *Given a random variable X with mean μ and variance σ^2, the sampling distribution of the sample mean ($\overline{x}$) follows a normal distribution with mean $\mu_{\overline{x}} = \mu$ and variance equal to $\sigma^2_{\overline{x}} = \frac{\sigma^2}{N}$ for a sufficiently large number of samples (N).*

Finally, we have simulated our algorithms where the position of the robots and the points of the new formation are calculated at random as was explained in Section

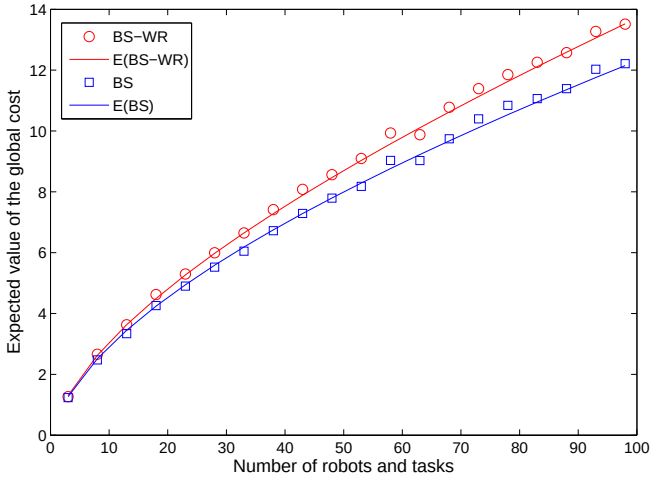


Figure 24: Expected value of the global cost over 100 simulations where the costs follow the distribution explained in Section 3.4.2.1. The circles represent the results from simulation for the BS-WR algorithm and the squares for the BS algorithm. The theoretical results, E(BS) and E(BS-WR) respectively, are shown as solid lines.

3.4.2. For the BS-WR algorithm we use Formula (15) to calculate E_{GC}. Since it is not possible to calculate an analytical solution, we used a numerical method to solve the integral. The trapezoid rule was used since the function, $F_X(x)$ (13), is not twice continuously differentiable and other methods, such as the Simpson's rule, have problems in finding an accurate solution for this case. On the other hand, for the BS algorithms we have used the formulae commented in Section 3.4.2.3 which has also been calculated using the trapezoid rule. In Figure 24, we observe that our theoretical results match the simulation ones. BS algorithm still obtains better results than the BS-WR algorithm and therefore, it can be said that reallocation improves the results in different scenarios.

In summary, we have shown that the probabilistic analysis models accurately our task allocation algorithms, and our results can be used to predict the behavior of these algorithms in different scenarios.

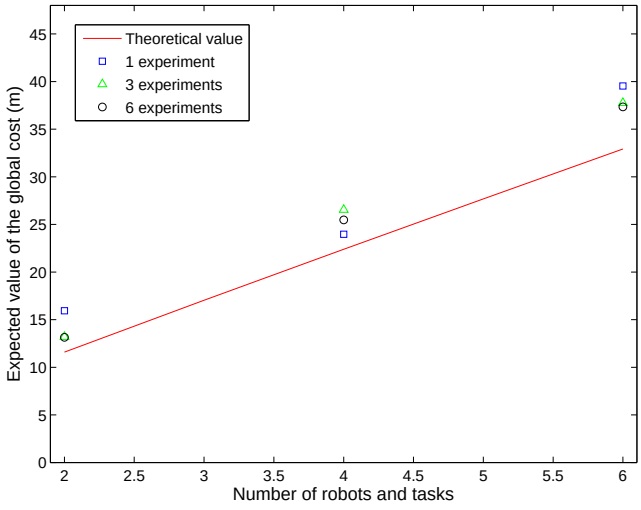


Figure 25: Results from the experiments for the dispersion scenario. The costs were uniformly distributed between $[5, 7]$ meters. The mean from different number of experiments have been calculated for 2, 4 and 6 robots.

3.7 Results from experiments with real robots

Thirty-six experiments have been run using our robotic testbed (see Section 2.5.1) with a testbed arena of 15x23m^2. The main objective of these experiments is to show that our theoretical results are still valid even when we use real robots, with all the noise and imperfections. We have implemented the BS algorithm since it is the most complete of the two and we have tested it in the two different scenarios (dispersion and random). First, we run the experiments emulating a dispersion scenario where the robots where uniformly distributed in an circle of radius one meter and the tasks in a doughnout with 6 and 7 meters as the inner and exterior radius respectively. Therefore, the costs are uniformly distributed between $[5, 7]$ meters. A method based on [14] was used to obtain points uniformly distributed on l_p doughnuts instead of l_p balls. As can be observed from Figure 25, the sample mean calculated from the experiments gets closer to the theoretical value when the number of experiments increases. Also, in Figure 26, it can be seen the same effect for the random scenario, where the robots and tasks have been calculated at random in the 15x23m^2 arena.

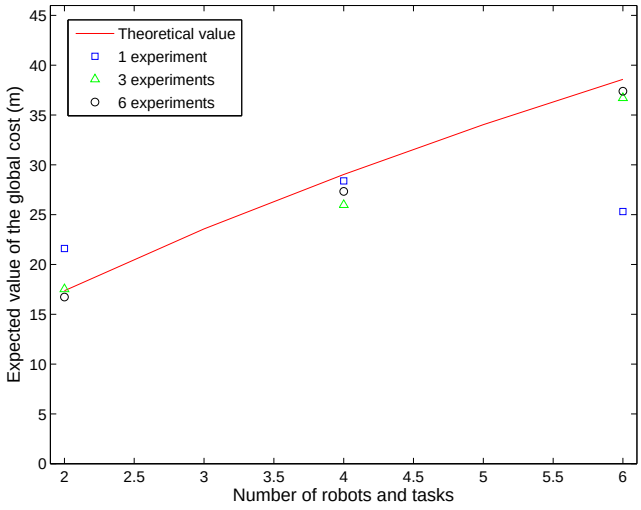


Figure 26: Results from the experiments for the random scenario. The position of the robots and points of the new formation were distributed as: X coordinate follows a uniform distribution between $[0, 15]$ meters and the Y coordinate between $[0, 23]$ meters. The mean from different number of experiments have been calculated for 2, 4 and 6 robots.

On the other hand, there are cases (4 robots in both scenarios) that the mean from the first experiment gets results closer to the theoretical value than with a higher number of experiments. This can be explained due to the random nature of the costs. The same can happen when you calculate the mean using the samples obtained from a random generator function.

Finally, we have demonstrated that our theoretical results are still valid when real robots are used and we have shown that these results accurately model the behavior of our algorithms. Thus, the results presented in this paper can be used to compare different algorithms in different situations and predict their behavior.

3.8 Conclusions

In this chapter a general probabilistic analysis for market-based algorithms that solve the Initial Formation Problem has been developed. This analysis consists of calculating the expected value of the global cost and we use this as a metric to compare

different algorithms. The analysis has been applied to two different algorithms in two different scenarios. The advantage of this approach, in comparison with a worst-case analysis, is that this metric provides a much better indication of real world results. The worst-case analysis is very pessimistic and it could happen that an algorithm performs better in real applications but has a lower worst-case performance.

Our algorithms have been tested in simulation and implemented on real robots. From our experiments, it can be said that our theoretical results are close to the ones obtained from experiments and therefore, our probabilistic analysis accurately models task allocation algorithms. Also, this analysis is completely general and can be used with other cost distributions or different scenarios. Thus, our analysis can be adapted to different situations and the results will more closely resemble real world results than those obtained from a general, situation-independent method. Since our metric is the expected value of a random variable, we have figured out how well our results are related to the ones from experiments dependending on the number of runs. We have calculated a formula that gives us an idea of that maximum difference.

Finally, we have extended our analysis to situations when the number of robots and tasks are not the same, and have shown that for situations with more robots than tasks both algorithms perform in a similar way and the global cost tends to a constant value instead of increasing with respect to the number of robots. On the other hand, when the number of robots is smaller than the number of tasks, the BS algorithm performs much better than than BS-WR algorithm thanks to the use of reallocation.

CHAPTER IV

CONSIDERATIONS FOR REAL WORLD APPLICATIONS

Regardless of the fact that the multi-robot task allocation problem has been studied for the last decade, most of the algorithms have been tested assuming idealistic simulations and only considering waypoint tasks where the cost is just the euclidean distance. Only a few have been tested on real robots [10, 18, 28, 85] and they usually deal with proving of concepts. For these reasons, it is important to test these algorithms while they are integrated in a complete robot architecture, taking into account the different aspects of a functional robot. Also, we think that is important to study the effects that real-world considerations, such as terrain knowledge, have on the efficiency of the distributed task allocation method for multi-robot systems.

The chapter is organized as follows. A modification of the presented algorithms is explained that reduces the number of messages keeping a similar efficiency in comparison with the optimal solution. The fault tolerance aspect is covered in the next section which uses a distributed algorithm to recover the tasks allocated to a robot even when it loses communication capabilities or has a complete failure. Next, the integration of task allocation, path planning and navigation within a complete robotic system will be explained. Simulation results that compare the algorithms in an obstacle-strewn world in which the task allocation algorithms are integrated with two different path planning algorithms are presented. These algorithms have also been implemented on a team of physical robots and the results from several experiments will be explained. Finally, we present how this work could be applied to mobile sensor networks for achieving scientific measurements in arctic environments.

4.1 Reduction of communication messages for large number of robots

One of the main problems with market-based approaches is that the number of messages used in the allocation algorithms increases with respect to the number of robots. Although all robots may be in communication range, it is not efficient to allow all of them to participate in each auction. In order to reduce the number of messages exchanged between robots, a basic modification can be applied to the algorithms commented in the previous section:

- If a robot has not won any task yet, the algorithm remains the same and the robot sends a bid to all the announced tasks.
- From the moment a robot wins a task, it only bids on the tasks that have a lower cost than the one already won.

As can be seen in Figure 27 (original and improved algorithms), the number of messages is reduced with this simple change in the algorithms. It can be observed that the difference with the original algorithm increases with respect to the number of robots and tasks, reducing the number of messages by 43% for 20 robots. Also, it is interesting to point out that for small number of robots and tasks (less than seven) the improvement does not have a considerable effect in the number of messages since the reduction is smaller than 10%. Therefore, these kinds of improvements start to have importance in the performance of the system when the number of robots is larger than 6 robots. This figure only shows the results for the BS algorithm, however this modification can easily be applied to the other RTMA algorithm with similar results. All the results presented in this section suppose that all the robots are in communication range.

However, this basic modification in the algorithms can be improved and we can reduce even more the number of messages when the number of robots is large enough.

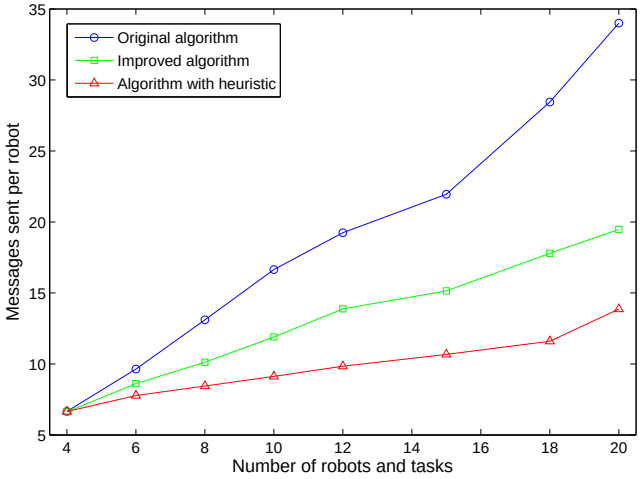


Figure 27: Comparison of the number of messages sent per robot among the original BS algorithm, the improved algorithm and the same algorithm with the improvements and the dynamic bid coverage radius heuristic algorithm. These results have been calculated as the mean over the same 100 random missions for both algorithms in a 1000mx1000m virtual world where each mission consists of going from an initial formation to a final one.

For that purpose we have developed a novel heuristic algorithm based on dynamic bid coverage radius. Each robot has a bid radius (b_r) value that is associated to each of the tasks that the robot is to announce. Only the robots within the bid radius, associated with the task position of the center of the circle, are allowed to bid on that task. This bid radius is not fixed and its value can change depending on the number of robots that have sent their bids. The objective is that the number of robots sending their bids will be equal to a desired number of bids (d_b) per auction. Therefore, the heuristic algorithm works as follows:

- At the beginning, the bid radius is large enough to cover the entire world, and therefore, all of robots are tasked to send their bids in the first auction. This initial value of the radius could be made smaller if we knew or had some idea of the distribution of tasks and robots, but for a general situation we think that considering all the robots for the first auction is a valid approach.

- As said before, each robot has its own bid radius value. This value is associated with the next task to be announced and, depending on the number of bids received, the following actions will be taken:
  - If the number of bids is smaller than a desired value, the bid radius will be increased for the next task to be announced using the following relation $b_r = b_r + \alpha * b_r$, where α is smaller than 1.
  - If the number of bids is bigger than a desired value, the bid radius will be decreased by a factor equal to β, i.e., $b_r = \frac{b_r}{\beta}$ where $\beta > 1$. As can be seen, the bid radius is decreased faster than increased since we want to reduce as fast as possible the number of robots that take part in the auctions, and therefore, minimize the messages used in the task allocation algorithm.

From our experiments, we have observed that it is better to keep the same bid radius when the number of bids are within $(d_b - 1, d_b + 1)$, where d_b is the desired number of bids. With the introduction of this "dead zone" in the bid radius, it reduces the number of times that the bid radius is changed and also it maximizes the number of times the number of robots that bids on the tasks achieves the desired value.

However, due to the special characteristics of the Initial Formation Problem each robot can only be allocated one task. This fact can result in the heuristic algorithm entering in a negotiation loop where one or more robots never take part in the different auctions, as can be seen in Figure 28. In this figure, robot 3 is really far from all the tasks and only two robots take part in the auctions. Therefore, the bid radius of robots 1 and 2 is increased until 2 robots (considering themselves) take part in the auctions. Thus, robot 3 will never be included in any of the bid radius and will not take part in any auctions and one of the three tasks will be reauctioned between the two other robots.

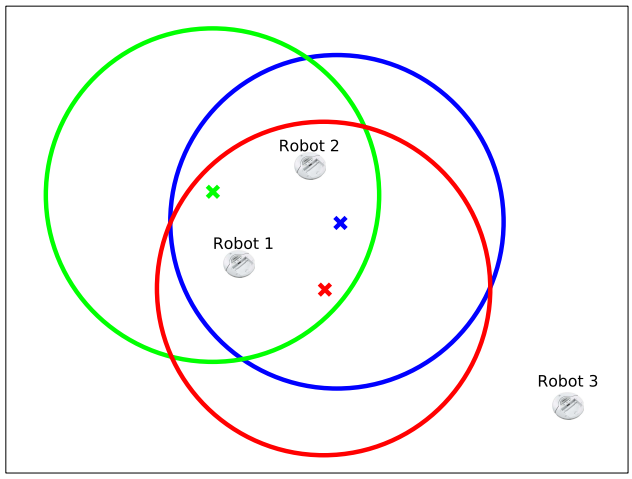


Figure 28: An example where only two of the three robots are within the task bid radii and since the desired number of robots is equal to two, the radii do not change and a loop in the negotiation is created, i.e., there will be 2 robots for 3 tasks and one task will be reauctioned all the time. The insertion of the robot radius is included in the heuristic algorithm to solve this problem.

In order to solve this problem, each robot has its own bid radius that increases everytime an announce message is received and a robot cannot bid on a task since it is out of the task bid radius. In this case, the robot bid radius is increased proportionally to the width or height of the world, i.e., $r_b = r_b + k * W$, where r_b is the robot bid radius, k is less than 1 and W is the largest value associated with the width and height of the world. With the use of the robot bid radius, the heuristic algorithm remains the same, but now, a robot bids on an auction when either the robot is within the task bid radius or the task is within the robot bid radius, solving the loop problem commented on previously.

Next, the improvements obtained with this heuristic algorithm will be presented and it will be demonstrated that the efficiency of the solutions remains the same. After several simulations, we have seen that the values of α, β and k with best results, for the type of missions we use, are: $\alpha = 0.5$, $\beta = 2.0$ and $k = 0.1$ with the desired number of bids equal to $min(0.5 * NumberOfRobots, 4)$. We have tested this heuristic algorithm with the same 100 random missions that were used for the original

and improved algorithm. These missions consist of going from one initial formation to a final one. The initial positions of the robots and the positions of the final formation are calculated at random using a uniform distribution in a 1000mx1000m world.

In Figure 27, we compare the number of messages derived from the original algorithm, the improved one and the same algorithm with the improvements and the dynamic bid coverage radius heuristic algorithm. It can be observed that the heuristicalgorithm reduces the number of messages sent per robot and the difference between the original and improved algorithm increases with respect to the number of robots and tasks, obtaining a reduction in the number of messages equal to 60% for 20 robots and tasks in comparison with the improved algorithm. Also, the heuristic algorithm only improves the performance in terms of number of messages when the number of robots and tasks is large enough (i.e. more than six).

Finally, it is important to demonstrate that the reduction in the number of messages does not result in a decrease in the efficiency of the obtained solutions. Table 3 documents the mean and the variance of the global cost (sum of the costs of each robot that executes a task) for the original, improved and improved with heuristic algorithms. As can be observed, the results remain very similar. Even with a significance reduction in the number of messages (60% for 20 robots and tasks between the original and the algorithm that uses the heuristic algorithm), the results are equivalent.

4.2 Complete fault tolerance algorithm for task allocation problems

One of the main reasons for solving the Initial Formation Problem in a distributed way is the capability to make the system highly fault tolerant without the need for a centralized control mechanism that may fail and stop the whole system from working correctly. However, making the task allocation algorithm distributed is not enough to overcome all of the different kinds of failures.

Table 3: Results computed for formations with different number of robots and tasks over the same 100 random simulations per each case in a 1000x1000m world. In each cell the mean and the variance of the global cost are presented. All the algorithms obtain very similar results. The heuristics applied to reduce the number of messages do not affect the efficiency of the solutions.

Tasks & Robots	***Original algorithm***	***Improved algorithm***	***Improved heuristic algorithm***
4	1473.52 (477.97)	1473.52 (477.97)	1473.52 (477.97)
6	2020.13 (482.02)	2020.55 (482.27)	2020.56 (482.27)
8	2443.57 (581.06)	2443.57 (581.06)	2443.95 (580.72)
10	2865.81 (651.57)	2867.87 (649.45)	2868.17 (651.98)
15	3749.97 (776.41)	3749.99 (776.41)	3756.32 (788.96)
20	4639.68 (916.49)	4639.85 (919.79)	4643.73 (937.32)

From a general point of view, there are three main types of robotic failures that can invalidate the task allocation algorithm to work correctly:

- Task failure: these are the failures that provoke a robot to not complete a task successfully. Some examples are: a broken camera that is unable to take images in a monitoring task, non-working motors, or a main failure in the obstacle avoidance sensors (such as a laser scanner).

- Communication failure: these failures are the ones related with the communication devices that do not allow the robots to communicate between each other.

- Total failure: these failures make robots stop and turn off immediately. These are usually related to energy problems, for example the loss of the main power generator.

We are supposing that a robot can detect both task and communication failures, and therefore, act accordingly. All distributed market-based task allocation algorithms can solve the fault tolerance problem, but only for task failures. In this case, the solution is really straightforward; if a robot cannot complete a task, it will stop

trying to execute it and reauction that task. This auction works in the same way as a regular one but the robot that could not finish the task will not take part in it. The task will be allocated to another robot that will finish the task. For the particular case of the Initial Formation Problem, the reauction of a non-completed task only makes sense if there are more robots than positions in the formation, i.e., there will be some idle robots that can take part in the reauction of the non-completed task. If there are the same number of robots and positions of the formation, the task will not be allocated to any other robot since only the robots that are not executing a task will take part in the auction. Also, it does not make sense to move one robot from one position of the formation to another one, since we are supposing that all the positions of the formation have the same priority.

As far as we know, there is no distributed task allocation algorithm that addresses fault tolerance algorithms which consider communication and total failures even when it is supposed that all the robots are in communication range. Our approach treats in the same way both types of failures. The only difference from the failed robot point of view is that in a communication failure, it could go to a base station to be repaired. But, in both cases, robots can no longer coordinate using our task allocation algorithms, and therefore, they are no longer useful from the coordination perspective.

Our approach to increase fault tolerance for the different failures is based on quorums [32] which is a technique used for data replication in distributed systems. The basic protocol was designed for distributed storage using clients and servers. Each client reads from r servers and writes to w servers. If $r + w > n$, where n is the total number of servers, then the intersection of every pair of read/write servers sets is non-empty, i.e., every read operation will see at least one copy of the latest value written. In order to know which of the read data is the latest, a time stamp (t) is associated with the data. The protocol works as follows:

- Read protocol for a data D:

  - Read $< D, t >$ from all the r servers.
  - Select D with the latest time stamp t.

- Write protocol for a data D:
  - Read value according to above protocol to determine current time stamp t.
  - Write $< D, t >$ to all w servers with time stamp $t' > t$.

We have adapted this protocol to a multi-robot system in order to recover the tasks allocated to any robot without the need for a central server that knows the allocations of all the tasks or the need to save all the tasks in all the robots. Therefore, our idea is that when a robot has a communication or total failure, we can use an adaptation of this protocol to recover the tasks allocated to that robot and then reauction them, so the mission will be finished successfully.

As all fault tolerance problems, there are two phases that have to be solved:

- Failure detection: when a robot has a communication or total failure and cannot warn the rest of the robots. In this case, we must implement a distributed protocol that will detect a failure in one of the robots. This protocol will be based on alive messages and it will be explained next.

- Task recovery: once a robot failure has been detected, an adaptation of the quorum algorithm will be used to recover the tasks allocated to the failed robot and reauction them again as was explained with the task failures.

The implemented failure detection algorithm is based on alive messages. Basically, every robot broadcasts every time T a message which means that the robot is working correctly. If during the wait time T, the robot sends another broadcast message such as an announce message of the task allocation protocol, the timer is reset, and

therefore, the number of messages to be sent is reduced. The other problem related to failure detection is determining which robot should report that a robot failure has occurred.

Our approach is based on the identification of all the robots by a unique identification number. Each robot is assigned a list of X number of robots whose identification number is bigger than itself (if the biggest identification number is used and the number of robots included is less than X, the list will be completed starting from the robot with the lowest identification number). The number X is usually a percentage of the total number of robots and its value is calculated taking into account the probability of failure of X number of robots. Therefore, the value of X will depend on the application. A sufficient value will be the one that makes the probability of X number of robots fails smaller than 0.1. Finally, each robot will look after the first alive robot on the list and, if it does not send an alive message during a time of period greater than T, the watchful robot will start the recovery procedure using the modified quorum algorithm explained next.

In a multi-robot system, there is not a distinction between clients and servers, so all the robots will work as client and servers. Also, the data to save includes the task or tasks allocated to a specific robot and only one robot will change its own data. Therefore, the time stamps associated with the data can be generated locally since the data associated to a robot will only be changed by itself.

Finally, the modified quorum algorithm works as follows:

- Every time there is a change in the list of tasks associated to a robot, i.e., a robot has won a new task or it has sold a task to another robot, it will transmit the current tasks to w number of robots with a time stamp generated locally.

- When a robot detects that another robot has failed, it will ask for the tasks associated to r number of robots. The data with the newest timestamp will

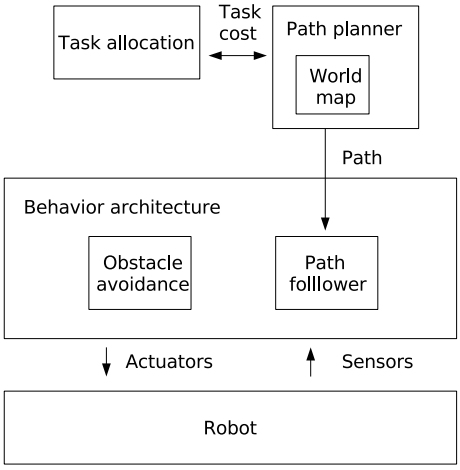


Figure 29: Scheme that shows the integration of a task allocation algorithm in a complete system ready to be used in a real world application. The path planning algorithm is used to calculate the cost of the tasks and as an input for the path follower algorithm which is combined with obstacle avoidance using the DAMN architecture.

be used to recover the tasks associated to the failed robot and they will be announced again.

4.3 Integration within a complete robotic system

We have integrated our task allocation algorithm, using a multi-robot architecture based on modules [52], within a complete robotic system ready to be used in real world applications. As can be seen in Figure 29, in each robot the task allocation algorithm has been integrated with a path planner algorithm and the execution of the tasks are within a behavior architecture that combines the path following algorithm with obstacle avoidance.

When the task allocation algorithm has to calculate the cost for a specific task, it sends the location data to the path planning module. Next, the path planner calculates the path using the information from an internal wold model and sends it back to the task allocation module as a list of points. We have used a 2D grid as the world model where each grid is considered as navigable or non-navigable.

During the negotiation process, it is possible that the task allocation algorithm

has to calculate several times the cost for the same task. For this reason, everytime the path planning module calculates the path for a task, it will save the path and its cost. In this way, we reduce the computation power and the calculation time for future requests. Each task is identified by a unique sequence number created by the scientist interface.

After all tasks have been allocated, each robot starts the execution of its own. The path planning module sends the path to the path follower module which is combined with an obstacle avoidance behavior. All three behaviors are combined using a DAMN architecture [64].

4.3.1 Realistic simulations

In order to prove that our algorithms still obtain the same kind of efficiency in real world applications, we have integrated our task allocation algorithms in a complete system ready to be used in real world applications within our multi-robot simulator. As was described in Section 4.3, in each robot the task allocation algorithm has been integrated with a path planner algorithm and the execution of the tasks are within a behavior architecture that combines the path following algorithm with obstacle avoidance.

Two of the most popular path planning algorithms have been implemented: A^* algorithm [58] and RRTs (Rapidly-exploring Random Trees) [48]. These allow the system to integrate map-based information in the task allocation scenarios. The first algorithm is based on a heuristic estimator to find the optimal solution faster than general search algorithms such as breadth-first or depth-first search. Even so, for robotic applications, the A^* algorithm still requires a significant amount of processing power, specially for large state spaces with constraints. RRTs is also a search algorithm that has a random nature and the quality of the solution cannot be determined a priori, but it is faster than A^*. This algorithm works like a search tree

that starts from an initial state and is expanded by performing incremental motions towards the direction of random points. The main difference between this algorithm and a random walk is that the latter suffers from a bias towards places already visited, while RRTs works in the opposite manner by being biased towards places not yet visited. Specifically, we have used the bias version of RRTs with a probability equal to 0.05.

For navigation, we have selected the DAMN architecture to combine the obstacle avoidance and path follower algorithms. This architecture was designed to combine different behaviors, specially, for mobile robots in unknown and dynamic environments which fits our demonstration scenario. Each of the behaviors votes for a set of possible actuators values satisfying its objectives. Then, an arbitrator combines those votes and generates actions which reflects the behaviors objectives and priorities. Regarding the behaviors, a laser scanner was used as the sensor for the obstacle avoidance and the Pure Pursuit algorithm [59] has been used as the path follower. The Pure Pursuit algorithm geometrically determines the curvature that will drive the vehicle to a chosen path point defined as one lookahead distance from the current position of the robot.

We have used Player/Gazebo [27] to simulate the environment and the robots. We will focus on a monitoring application where robots have to navigate towards some specific locations and take environmental measurements. As will be seen in the next section, we will use the iRobot Create platform to test our algorithms. For that reason, we have created a 3D model of those robots to be used in the simulator (see Figure 30).

We are interested in the effect that obstacles density have on the performance of our task allocation algorithms and whether differences depend on the path planner algorithm that has been used. We made a classification based on the percentage of non-navigable terrain (in this case obstacles): high navigable terrains (less than

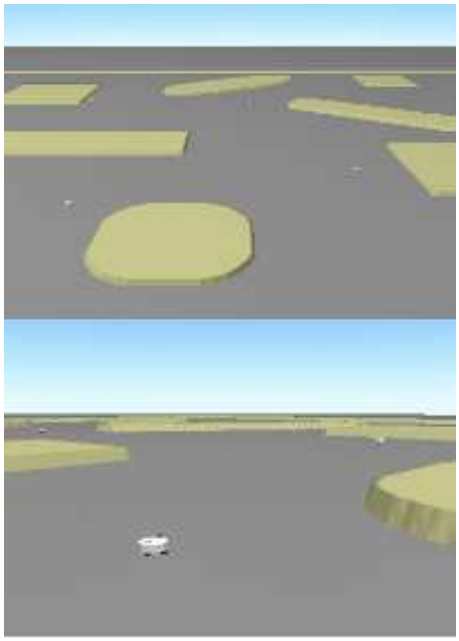


Figure 30: Snapshots of the simulator Player/Gazebo. At the top, an aerial view of the environment with obstacles. At the bottom, a close view of the 3D model of our test platform.

15% of non-navigable terrrain), medium navigable terrain (between 15% and 30% of non-navigable terrain) and low navigable terrain (more than 30% of non-navigable terrain).

For our simulations, we have used three different scenarios, all of them with a 75x75 m^2 area, to test our complete robotic system for this type of application. The first scenario (see Figure 31) has 5% of non-navigable terrain. The second scenario has 20% of non-navigable terrain (see Figure 32), and the last scenario has 40% of non-navigable terrain (see Figure 33). Also, Figures 32 and 33 show the solution obtained using our algorithms and the path followed by the robots using the RRTs and A^* algorithms respectively. It can be observed directly how the A^* obtains the optimal path while the RRTs has a lower rate of finding a path close to the optimal one. This fact will have a large impact on the performance of the task allocation algorithms, as will be commented next. Also it is interesting to observe that in Figure 32, one robot is forced to navigate to the furthest task for itself due to the outcome from the BS

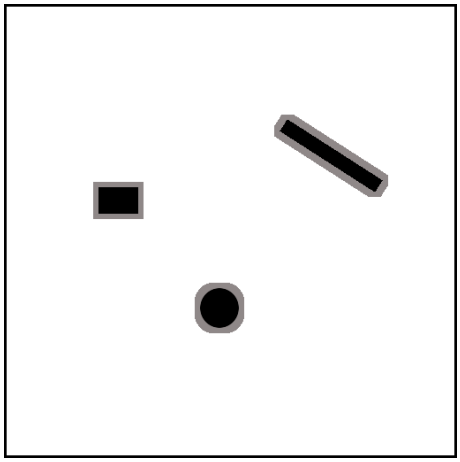


Figure 31: Scenario with 5% of non-navigable terrain. The obstacles are increased virtually the size of the robot, so they do not navigate too close to them. This reduce the probability of a collision due to noises and inaccuracies in the sensors and the map.

algorithm.

Only 20 simulations have been run per case (due to the complexity of these simulations) where the position of the robots and tasks have been calculated at random (avoiding the areas considered obstacles in the world). We first ran our simulations using the A^* for path planning. The results obtained from these simulations are showed in Table 4 where each cell represents the mean of the global cost over 20 missions, i.e., the sum of the distance traveled by all the robots, and the error in percentage with the optimal solution. It can be seen that the RTMA algorithm still obtains better results than the BS algorithm when it is integrated in a complete robotic system. Also, the results obtained with the complete system are equivalent, in comparison with the optimal solution, to the results obtained in Section 2.4. The improvements obtained with RTMA, in comparison with the BS algorithm, are of the same order of magnitude and both algorithms obtain similar results in all the scenarios. Therefore, the integration of our task allocation algorithms in a complete robotic system, with the A^* planner, does not affect the task allocation algorithms performance.

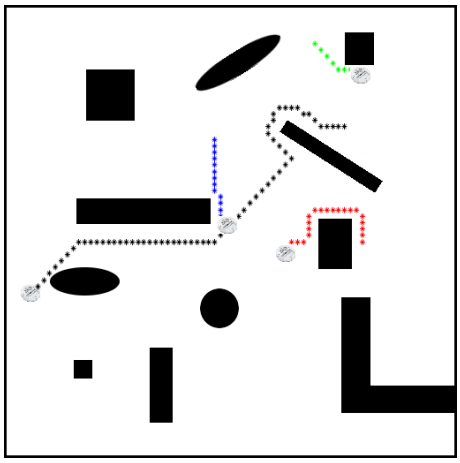


Figure 32: Scenario with 20% of non-navigable terrain. The paths show the solution of one of the random missions obtained using the BS task allocation with the A^* path planner.

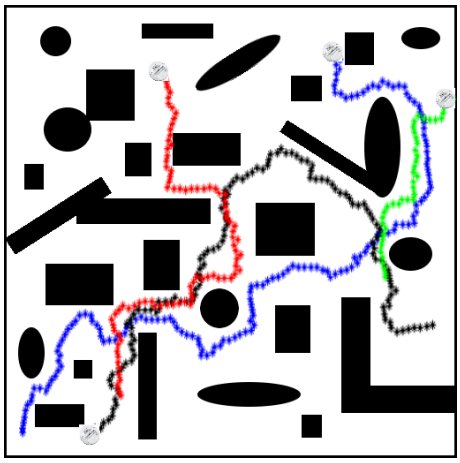


Figure 33: Scenario with 40% of non-navigable terrain. The paths show the solution of one of the random missions obtained using the RTMA task allocation with the RRTs path planner.

Table 4: Results computed for formations with different number of robots, tasks and obstacles over 20 simulations per each case using the A^* algorithm. Each cell represents the mean of the global cost and the mean error in percentage in comparison with the optimal solution. The obstacles are distributed as can be seen in Figure 31 for the 5% scenario, in Figure 32 for the 20% scenario and in Figure 33 for the 40% scenario.

Tasks & Robots	***Scenario***	***BS***		***RTMA***		***Optimum***
		Mean	Error	Mean	Error	
4	5%	124.74	4.58%	119.99	0.60%	119.27
4	20%	127.38	8.70%	119.62	2.07%	117.19
4	40%	161.94	6.85%	153.61	1.35%	151.55
6	5%	144.02	6.54%	139.88	3.47%	135.18
6	20%	187.76	7.53%	177.25	1.51%	174.60
6	40%	216.88	7.08%	204.92	1.17%	202.54
8	5%	197.52	8.94%	186.72	2.98%	181.31
8	20%	208.08	9.16%	195.86	2.75%	190.61
8	40%	261.62	12.63%	235.74	1.48%	232.29

The same random missions, for each scenario, have been used for the optimal solution and the two task allocation algorithms. The optimal solution has been calculated using the A^* algorithm with the Hungarian method [43], i.e., all the different optimal paths between every robot and task have been calculated using the A^* algorithm, then the distance of all these paths have been used as the values of the cost matrix that represents the task allocation problem as a job assignment problem. Finally, the Hungarian method has been applied to that cost matrix to calculate the optimal assignment.

Next, we tested our task allocation algorithms with the RRTs instead of the A^* algorithm. The results are shown in Table 5. First, it can be observed that these results are worse than using the A^* algorithm. This makes sense since the RRTs algorithm does not ensure any kind of efficiency of the solution. Also, when RRTs are used, the differences between both algorithms decreases and there is even one case where the BS algorithm performs a little bit better than the RTMA algorithm.

Table 5: Results computed for formations with different number of robots, tasks and obstacles over 20 simulations per each case using the RRTs algorithm. Each cell represents the mean of the global cost and the mean error in percentage in comparison with the optimal solution. The obstacles are distributed as can be seen in Figure 31 for the 5% scenario, in Figure 32 for the 20% scenario and in Figure 33 for the 40% scenario.

Tasks & Robots	***Scenario***	***BS***		***RTMA***		***Optimum***
		Mean	Error	Mean	Error	
4	5%	159.68	33.88%	155.91	30.72%	119.27
4	20%	155.49	32.68%	149.92	27.92%	117.19
4	40%	190.78	31.98%	186.34	29.00%	144.55
6	5%	172.02	27.25%	171.33	26.74%	135.18
6	20%	235.70	34.99%	223.96	28.27%	174.60
6	40%	290.44	43.39%	291.62	43.98%	202.54
8	5%	242.09	33.52%	229.44	26.54%	181.31
8	20%	258.40	35.56%	252.77	32.61%	190.61
8	40%	354.56	52.66%	345.69	48.82%	232.29

In summary, it has been shown that the performance of the task allocation algorithms is better with the A^* algorithm rather than RRTs. Also, the use of RRTs reduce the advantages obtained with a more complex algorithm, such as the RTMA algorithm, and make the results of both algorithms very similar. The percentage of non-navigable terrain in the scenario seems to not affect the performance of the system and similar results have been obtained for the three different scenarios.

Finally, for this kind of application, where robots use an occupancy grid to navigate in 2D, the computational complexity of A^* is not high, and therefore, it is the best option. However, the RRTs algorithm should not be completely discarded since A^* might be too slow to be applied in some scenarios with high dimensional state spaces with constraints.

4.4 Experimental results with real robots

The main objective of these experiments is to show that our simulation results are still valid even when we use real robots, with all the noise and imperfections. The

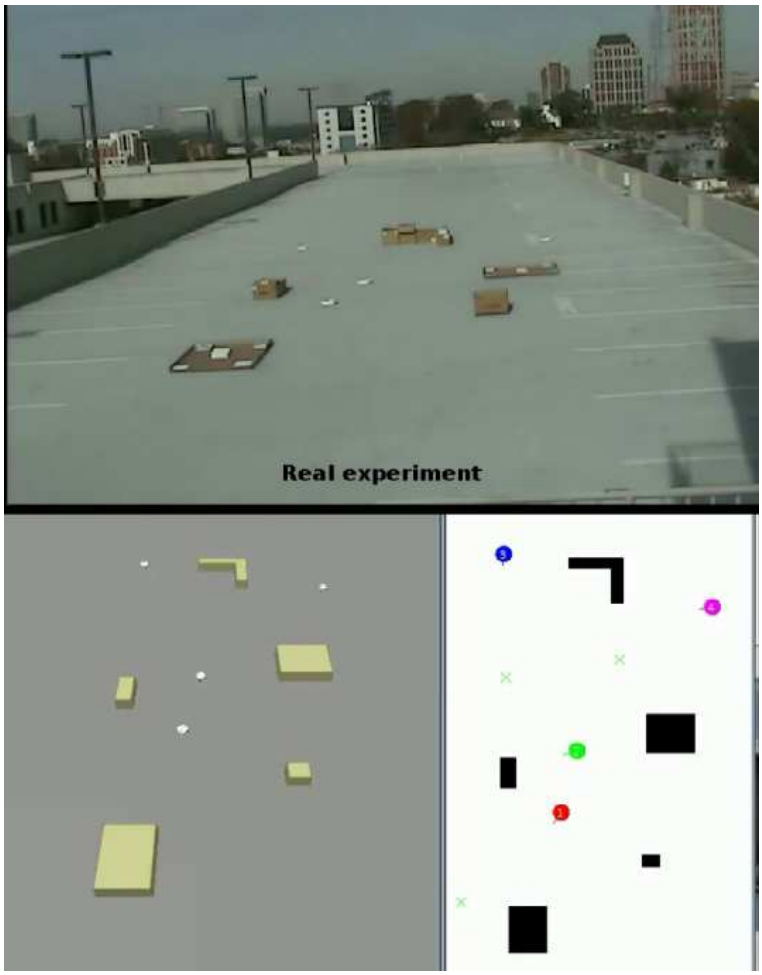


Figure 34: Team of robots running one of the experiments in an arena of 15x23m^2 with obstacles. Visual interface used to follow the experiments.

testbed described in Section 2.5.1 was used for the following experiments. The only difference is that a 15x23m^2 arena with obstacles was used instead.

Since the best results are obtained with the A^* algorithm, only experiments with this path planner were performed. Different number of robots were considered. Specifically, four experiments have been run with two robots, six with four robots and eight with six robots. In total, 18 experiments have been run with each of the two algorithms. All these experiments have been performed in an 15x23m^2 arena where the positions of the robots and tasks have been calculated at random avoiding the areas with obstacles.

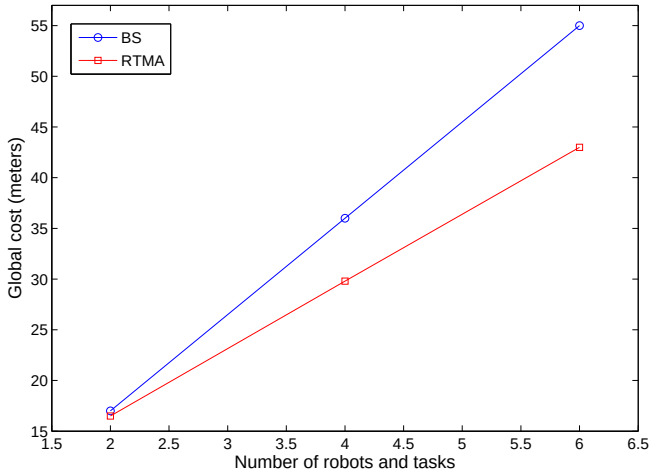


Figure 35: Results from the experiments in an arena of 15x23m^2 with obstacles (mean of the global cost). The BS and RTMA algorithms have been tested with 2, 4 and 6 real robots integrated in a complete robotic system including the A^* algorithm as path planner.

In Figure 35, the results from the experiments are shown. It can be seen that these results follow the same dynamic as the simulation results where the difference between both algorithms increases with respect to the number of robots.

4.5 Application to mobile sensor networks for achieving scientific measurements in arctic environments

Recently, it has been discovered that the giant ice sheets covering Greenland and Antarctica have been shrinking at an accelerated rate. While it is believed that these regions hold important information related to global climate change, there is still insufficient data to be able to accurately predict the future behavior of those ice sheets. Satellites have been able to map the ice sheet elevations with increasing accuracy, but data about general weather conditions (i.e. wind speed, barometric pressure, etc.) must be measured at the surface. A mobile, reconfigurable sensor network would allow the collection of this data without the expense or danger of human presence. For this to be a viable solution though, a method must be developed to allow multiple robotic scientific explorers to navigate these arctic environments. Specific

technological achievements that must be addressed for deployment of this surface-based mobile science network include estimating terrain characteristics of the arctic environment, incorporating these characteristics into a robot navigation scheme, and using this scheme to deploy multiple robotic scientific explorers to specific science sites of interest. In this chapter, we discuss an infrastructure that addresses these issues in order to enable successful deployment of these robotic scientific explorers.

Although weather data from glacial regions is considered important and valuable, this data is difficult to obtain. Currently, human expeditions must be sent to collect this data, which is costly, time consuming, and dangerous. Yet, this approach yields data about a very limited area, and covers only a short duration in time. To help alleviate this issue, a set of fixed weather stations have been installed, known as the Greenland Climate Network. While these weather stations provide a continuous data feed, with only 18 such stations covering an area of over 650,000 sq. mi, the spatial resolution is still very course.

In contrast, a group of autonomous mobile weather stations could be deployed in these regions. This would allow scientists to gather arbitrarily dense weather data about the area of their choosing, all while staying safely within the arctic outpost. In order to achieve such goals, several technological achievements must first be addressed. Namely, a mobile platform capable of traversing arctic terrain must be developed, a means of assessing environmental hazards must be added to the navigation system, an intuitive interface must be design to allow scientists to command the rovers' position and formation, and the rovers must be able to automatically reassign tasks between themselves in the event of failure.

4.5.1 Robot Platform

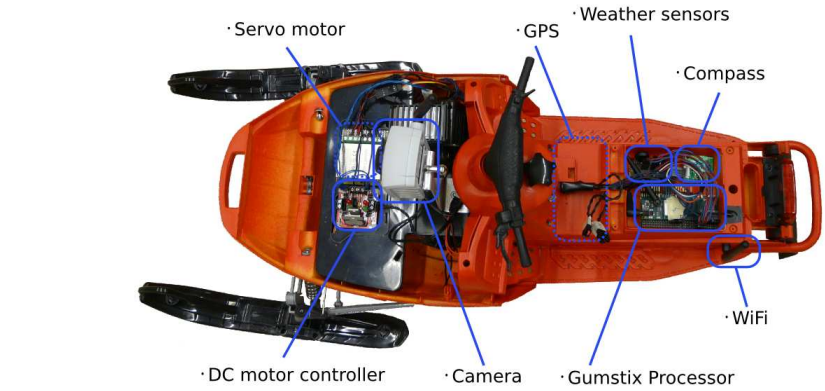
Despite being covered by snow, arctic regions present a large assortment of terrain challenges. Large quantities of fresh surface snow can be present during certain times

of the year. This fresh snow is soft, creating a potential sinking hazard for wheeled vehicles. Over time the winds harden the snow surface making it more amenable to locomotion. However, these same winds also sculpt the snow into dune-like structures that can be as large as one meter, again impeding motion. Tracked vehicles have been developed to overcome the specific challenges of arctic travel. The most famous of these devices is the snowmobile, but other variations exist ranging in size from small single person vehicles to bus-sized multi-passenger coaches. These platforms have been successful in military, commercial, and science applications since their development in the 1940s.

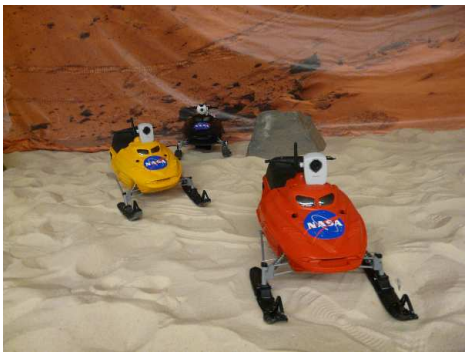
For these reasons a snowmobile chassis was selected as the base for the "Arctic Crawler"[1] prototype mobile sensor. A set of three prototype robotic rovers were constructed in our lab in anticipation of field testing. The rovers are based on an RC snowmobile chassis and have been retro-fitted with a Connex 400XM processor from Gumstix. This motherboard contains a 400MHz ARM processor, wireless 802.11g ethernet, and bluetooth capabilities. Additionally, a Robostix board was added, which includes an Atmel ATMega 128 RISC microcontroller, providing both SPI and I2C serial ports, general purpose IO pins, PWM outputs, and an ADC unit. The original steering mechanism was replaced by a servo motor to provide proportional steering control, while an H-bridge amplifier provides modulated voltage to the DC drive motor for variable speed control. Both motors are controlled by the Gumstix processor using the PWM outputs.

For navigation, a GPS unit connects to the embedded processor via the bluetooth interface, while a magnetic compass provides heading information via the I2C serial bus. Sensor data and internal state information are exchanged between scientists and other rovers over the bidirectional wifi link. Additionally, a 0.3 Megapixel wireless camera on-board each Arctic Crawler provides real-time images.

[1]Pictures and information about the robot platform courtesy of Stephen Williams



(a)



(b)

(c)

Figure 36: (a) A diagram illustrating the internal electronic components of the Arctic Crawler rovers. Images (b) and (c) show the fully assembled rovers in a lab setting and at a test site near Cleveland, Ohio respectively.

To simulate the science objectives of the mobile sensor network, a weather-oriented sensor suite was added to each rover. Ultimately this science package will include an anemometer and a solar radiation sensor, among others. However, the size and expense of these types of sensors were not a good fit with the small footprint of the prototype platform. Instead, a set of solid-state sensors were selected that could measure meaningful weather related data and still fit within the confines of the rover's chassis. The final instrument suite includes sensors to measure temperature, barometric pressure, and relative humidity. Figure 36 shows a diagram of the internal robot components, as well as images of the fully assembled rovers.

4.5.1.1 Navigation scheme

The architecture explained in Section 4.3 has also been used for this robotic platform. The only difference is that a slope assessment algorithm has been incorporated as a new behavior to avoid robots of roll-over despite the robustness provided by the snowmobile chassis. Therefore, in each robot, a path planning algorithm, an obstacle avoidance routine, a slope assessment algorithm, and a task execution unit have all been integrated into a single behavior-based architecture. Navigation is implemented using the DAMN architecture to combine the competing outputs of each behavior module. To minimize the likelihood of roll-over, a fuzzy logic slope assessment scheme has been developed to keep the rover on level terrain [62]. The behavior makes use of a slope estimation technique using only a single camera described in [80].

4.5.2 Scientist interface

Once the rovers can successfully navigate within the arctic environment, a mean of sending command positions must be created. A simple graphical interface has been developed that allows scientists to specify the desired sensor measurement locations. The main window of this interface is an aerial view of the terrain, as shown in Figure 37, where scientists can see the current location of the robots and the specified positions. A log window allows all the different actions taken by the robots to be viewed. For example, the current state of each task and to which robot it is assigned may be easily assessed via the log window. Also, this information is saved with an associated timestamp for each action for later review.

A menu window is available to configure and select the different options of the GUI. One such option is the source of destination information for the team of robots. This can be specified graphically using the map of the environment or with a plain-text mission file (see Figure 38), which is useful for offline mission planning. The tasks can be sent directly to a specific robot or our distributed algorithms can take

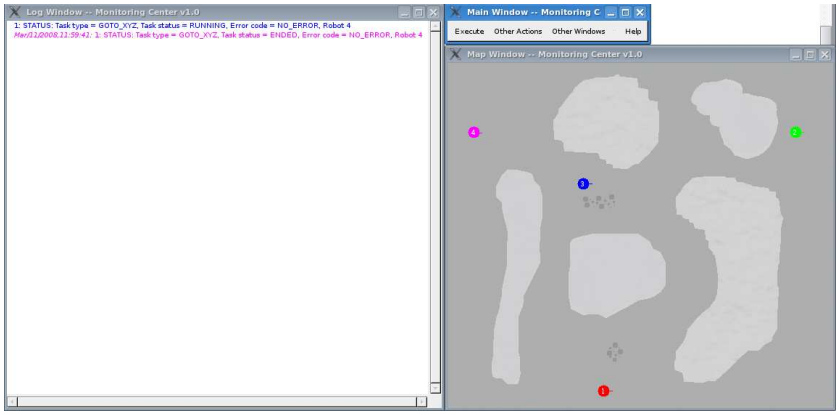


Figure 37: On the left, the log window with all the information related to one of the tasks. On the right, an aerial view of the terrain with the current positions of the robots.

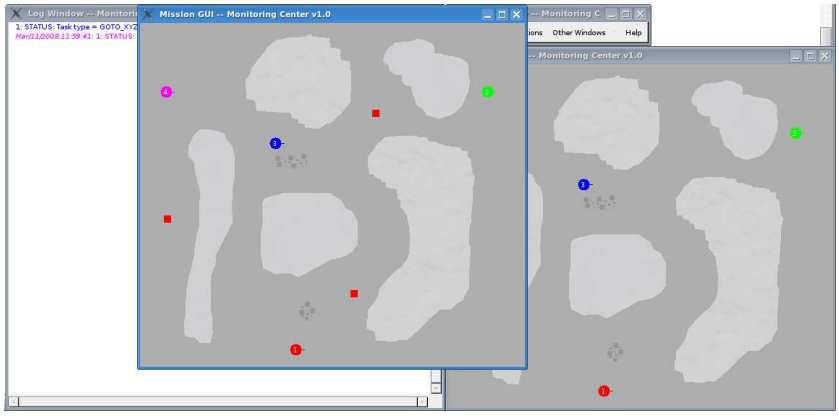


Figure 38: Graphical interface to specify the locations to be sent to the team of robots. Red squares represent the locations from which the robots will take environmental measurements.

care of the task allocation. Future work includes the possibility to show, in real time, the weather related data taken by each of the robots.

4.5.3 Simulation environment and results

Expeditions to the arctic are time consuming and expensive. Consequently, having a realistic simulation environment in which to test various algorithms is beneficial. Since we wanted a simulation as realistic as possible, a $200m$x$200m$ arctic scenario has been designed in Gazebo [27] and a replica of our snowmobile robot has been modeled as shown in Figure 39. Also, errors in the localization sensors have been introduced (GPS

Figure 39: Snapshot of the simulator Gazebo in a scenario that simulates an arctic terrain with obstacles. Our robot, a snowmobile, equipped with a laser scan sensor. Three different kinds of non-navigable terrain are shown: hills with high slope on the left, rocks on the right and ice layer in the middle.

and odometry) and a Kalman Filter developed to reduce the uncertainty. Since the dimensions of our robots are small ($0.6m$x$0.3m$x$0.4m$), we think that our simulated world is large enough to run missions of the same order of magnitude as in the real world. As discussed in [35], not only obstacles can be considered as non-navigable part of the terrain. For example, in an arctic scenario we can find ice layers that are untraversable due to the risk of breakage. We have considered three kinds of non-navigable terrains (see Figure 40): high slope hills, ice layers and terrain with middle or big size rocks. These non-navigable zones are considered by the robot as obstacles (see Figure 41). We have used an occupancy grid to represent the environment with a cell size of $1m$x$1m$ which will be used by the path planner algorithms.

We have used two different scenarios to test our complete robotic system for this type of application. The first scenario, see Figure 42, has a lower ratio between non-navigable and navigable terrain (around 20% of non-navigable terrain). On the other hand, the second scenario has a higher ratio (see Figure 43), more than 40% of non-navigable terrain. Also, Figures 42 and 43 show the solution obtained using the BS algorithm and the path followed by the robots using the RRTs and A^* algorithms

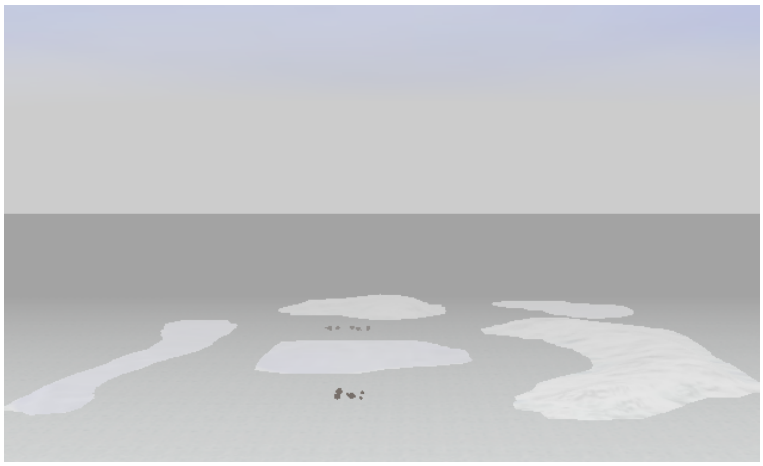


Figure 40: An aerial view of the simulated world where the three kinds of non-navigable terrain (ice layers, high slope hills and terrain with middle or big size rocks) can be distinguished.

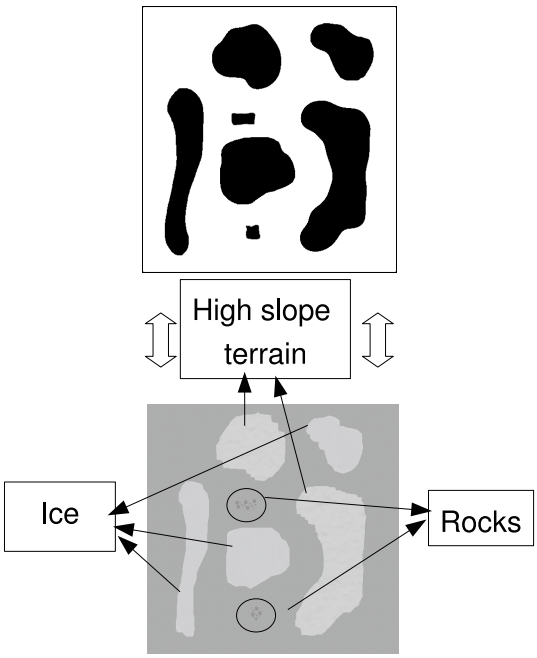


Figure 41: A 2D view of the simulated world. The different non-navigable areas are translated into obstacles in the occupancy grid that is used by the A^* and RRTs algorithms.

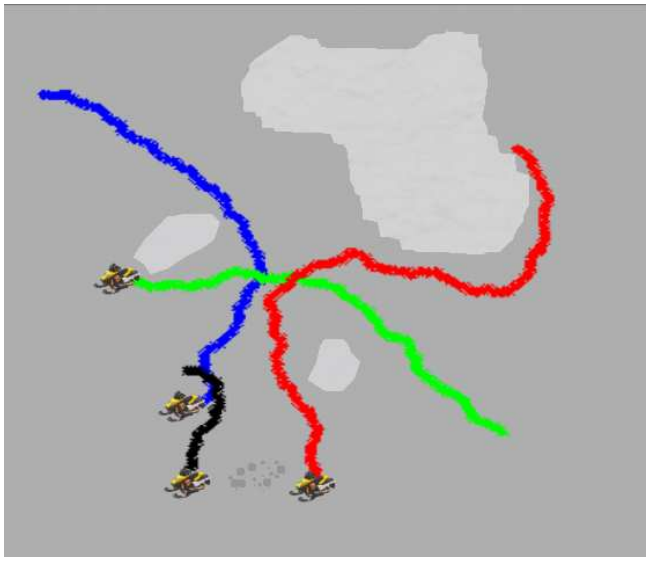


Figure 42: Scenario with a low ratio between non-navigable and navigable terrain. The paths show the solution of one of the random missions obtained using the BS task allocation with the RRTs algorithm.

respectively. It can be observed directly how the A^* always obtains the optimal path while the RRTs has a lower rate of finding a path close to the optimal one. This fact will have a large impact on the performance of the task allocation algorithms, as will be commented next.

Due to the complexity of these simulations, only 20 simulations have been run per case where the position of the robots and tasks have been calculated at random (avoiding the areas considered obstacles in the world). We first ran our simulations using the A^* for path planning. The results obtained from these simulations are showed in Table 6 where each cell represents the mean of the standard deviation of the global cost and the error in percentage with the optimal solution. It can be seen that the RTMA algorithm still obtains better results than the BS algorithm when it is integrated in a complete robotic system in arctic environments. The improvements obtained with RTMA, in comparison with the BS algorithm, are of the same order of magnitude for both scenarios. Therefore, the use of map-based information, with the A^* planner, does not affect the task allocation algorithms performance.

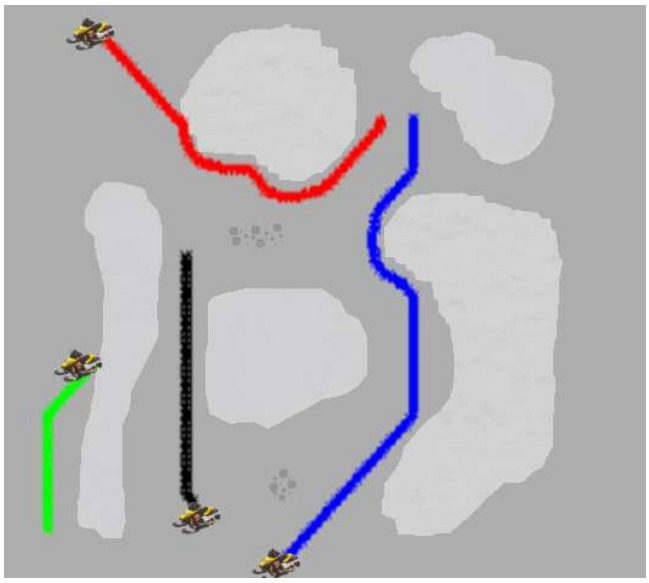


Figure 43: Scenario with a high ratio between non-navigable and navigable terrain. The paths show the solution of one of the random missions obtained using the BS task allocation with the A^* algorithm.

Table 6: Results computed for formations with different number of robots, tasks and obstacles over 20 simulations per each case using the A^* algorithm. Each cell represents the mean of the global cost, the standard deviation within brackets and the error in percentage in comparison with the optimal solution. The obstacles are distributed as can be seen in Figure 42 for the L scenario and in 43 for the H scenario.

Tasks & Robots	***Scenario***	***BS***		***RTMA***		***Optimum***
		Std	Error	Std	Error	
4	*L*	108.04	5.05%	106.54	1.14%	350.49
4	*H*	140.13	8.05%	123.02	0.7%	384.49
6	*L*	156.63	7.50%	138.43	1.99%	409.33
6	*H*	155.13	8.36%	142.14	3.32%	518.05
8	*L*	136.95	13.58%	119.57	3.16%	504.15
8	*H*	163.72	10.84%	155.41	4.57%	622.26

Table 7: Results computed for formations with different number of robots, tasks and obstacles over 20 simulations per each case using the RRTs algorithm. Each cell represents the mean of the global cost, the standard deviation within brackets and the error in percentage in comparison with the optimal solution. The obstacles are distributed as can be seen in Figure 42 for the L scenario and in 43 for the H scenario.

Tasks & Robots	***Scenario***	***BS***		***RTMA***		***Optimum***
		Std	Error	Std	Error	
4	*L*	132.62	26.73%	130.65	22.24%	350.49
4	*H*	231.26	28.68%	201.08	23.41%	385.49
6	*L*	205.08	34.02%	182.25	27.64%	409.33
6	*H*	196.32	36.52%	225.11	29.38%	518.05
8	*L*	224.37	43.16%	202.14	39.10%	504.15
8	*H*	234.79	33.82%	218.62	28.62%	622.26

The same random missions, for each scenario, have been used for the optimal solution and the two task allocation algorithms. The optimal solution has been calculated using the method explained in Section 4.3.1. Next, we tested our task allocation algorithms with the RRTs instead of the A^* algorithm. The results are shown in Table 7. First, it can be observed that these results are worse than using the A^* algorithm. Also, the standard deviations are higher than in the previous case. This makes sense since the RRTs algorithm does not ensure any kind of efficiency of the solution. Finally, it can be observed that the differences between both algorithms (BS and RTMA) decreases and the performance of both algorithms is very similar.

In summary, it has been shown that the performance of the task allocation algorithms is better with the A^* algorithm rather than RRTs. The results are similar to the ones obtained in Section 4.3.1 where the H scenario can be considered as a low navigable terrain (40%) and the L scenario as a medium navigable terrain (20%).

4.6 Conclusions

In this chapter different aspects related to a real world implementation have been considered. First, a modification of the task allocation algorithm that reduces the number of messages when the number of robots is large has been explained. This

algorithm is based on a adaptive bid radius that determines the number of robots that takes part in an auction. Then, a complete and distributed fault tolerance algorithm has been explained. This algorithm is able to recover the assigned tasks even when the robot is lack of communication capabilities or there is a total failure such as power loss.

The task allocation algorithms have been integrated in a complete robotic system considering the planning and execution levels of the tasks. This integrated system has been tested both in simulation and with real robots. Finally, it has been explained how the task allocation algorithms explained in this thesis can be used to help scientists to collect useful data and understand climate change causes using a mobile robotic sensor network.

CHAPTER V

CONCLUSIONS AND FUTURE WORK

This chapter presents the thesis conclusions and future developments. A summary of the main contributions of the thesis and an analysis of the objective achieved are firstly described. Then, future research activities to extend the work presented in this document are detailed.

5.1 Summary of contributions

The aim of this thesis is to develop a distributed algorithm that solves the Initial Formation Problem. This algorithm has to be fault tolerance and obtain solutions close to the optimal. A market-based approach has been used since it offers a good compromise between communication requirements and the quality of the solution.

In this thesis five different algorithms has been developed to solve the Initial Formation Problem in a distributed way. The first two (BS-WR and BS) are just a adaptation of the market approach to the Initial Formation Problem. However, the other three are an original work and it has been proven that they obtain better results than the basic algorithms. The algorithms have been extensively tested in simulation and with experiments using real robots.

Although the performance of the different algorithms have been evaluated in simulation, it is always more interesting to have a theoretical result that gives you an idea of your algorithm performance. One of the main contributions of this thesis is a general probabilistic analysis. This analysis can be applied to any of the algorithms and different scenarios. The work presented in this thesis is unique in the sense that it calculates the expected value of the global cost that can be seen as a performance metric. The advantage of this approach, in comparison with a worst-case analysis, is that

this metric provides a much better indication of real world results. The worst-case analysis is very pessimistic and it could happen that an algorithm performs better in real applications but has a lower worst-case performance. All the theoretical results have been validated with simulations and experiments.

Additionally, real world scenarios have been considered. First, a new heuristic algorithm has been described to reduce the number of messages used in the task allocation algorithms, specially when the number of robots is large. It has been proven that our heuristic algorithm reduces the number of messages considerably keeping the same efficiency of the solutions. Also, a novel fault tolerance algorithm was explained which allows the system to recover from any kind of failure and reauction the lost tasks, and therefore, successfully complete the mission. Simulations in a realistic environment have been presented where the task allocation algorithm has been integrated with a path planning algorithm and the execution of the tasks are within a behavior architecture that combines a path following algorithm with obstacle avoidance. Results from experiments with real robots considering obstacles have been used to prove that our algorithms work in noisy and realistic environments. Finally, it is explained how the work presented in this thesis has been applied to mobile sensor networks for achieving scientific measurements in arctic environments.

5.2 Future work

Next research will be driven to develop task allocation algorithms that only use local information but keep the same level of the solution efficiency. Another option could be to develop allocation algorithms that force communication coverage between robots during the execution of the allocated tasks.

Also, it could be interesting to adapt some of the ideas developed in this thesis to the market-based algorithms that already solve the general task allocation algorithm. In addition, the probabilistic analysis could be applied to those algorithms to obtain

theoretical results that describe their behavior.

Finally, we plan to perform field experiments using our snowmobile robots in arctic environments and we will seek to study the effect of mobile obstacles on the efficiency of the task allocation algorithms.

REFERENCES

[1] Agassounon, W. and Martinoli, A., "Efficiency and robustness of threshold-based distributed allocation algorithms in multi-agent systems," in *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems*, (Bologna, Italy), pp. 1090–1097, 2002.

[2] Alighanbari, M. How, J. P., "Decentralized task assignment for unmanned aerial vehicles," in *Proceedings of the IEEE Conference on Decision and Control and European Control Conference*, (Seville, Spain), 2005.

[3] Alighanbari, Mehdi How, J. P., "A robust approach to the UAV task assignment problem," *International Journal of Robust and Nonlinear Control, Special Issue on Cooperative of Unmanned Aerial Vehicles*, vol. 18, no. 2, pp. 118–134, 2008.

[4] Applegate, D., Cook, W., Dash, S., and Rohe, A., "Solution of a min-max vehicle routing problem," *INFORMS Journal on computing*, vol. 14, no. 2, pp. 132–143, 2002.

[5] Atay, N. and Bayazit, B., "Emergent task allocation for mobile robots," in *Proceedings of Robotics: Science and Systems*, (Atlanta, USA), 2007.

[6] Atay, N. and Bayazit, B., "Emergent task allocation for mobile robots through intentions and directives," tech. rep., Dept. of Computer Science and Engineering, Washington University in St. Louis, St. Louis, USA, 2007.

[7] Beard, R. W., McLain, T. W., Nelson, D. N., Kingston, D., and Johanson, D., "Decentralized cooperative aerial surveillance using fixed-wing miniature UAVs," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1306–1324, 2006.

[8] Bertsekas, D. P., "A new algorithm for the assignment problem," *Mathematical Programming*, vol. 21, pp. 152–171, December 1981.

[9] Bethke, B., Valenti, M., and How, J. P., "UAV task assigment," *IEEE Robotics and Automation Magazine*, vol. 15, no. 1, pp. 39–44, 2008.

[10] Botelho, S. C. and Alami, R., "M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, (Detroit, USA), 1999.

[11] Botelho, S. and Alami, R., "Multi-robot cooperation through the common use of "mechanisms"," in *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (Maui, USA), pp. 375–380, 2001.

[12] Brooks, R. A., "A robust layered control system for a mobile robot," *IEEE Journal of Robotics and Automation*, vol. RA-2, no. 1, pp. 14–23, 1986.

[13] Brumitt, B. and Stenz, A., "Grammps: A generalized mission planner for multiple mobile robots," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2, (Leuven, Belgium), pp. 1564–1571, 1998.

[14] Calafiore, G., Dabbene, F., and Tempo, R., "Uniform sample generation in l_p balls for probabilistic robustness analysis," in *Proceedings IEEE Conference on Decision and Control*, (Tampa, USA), pp. 3335–3340, 1998.

[15] Caloud, P., Choi, W., Latombe, J., Le Pape, C., and Yim, M., "Indoor automation with many mobile robots," in *Proceedings of the IEEE International Workshop on Intelligent Robotics and Systems (IROS)*, vol. 1, (Ibaraki, Japan), pp. 67–72, 1990.

[16] Dias, M. B. and Stenz, A., "Opportunistic optimization for market-based multirobot control," in *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (Lausanne, Switzwerland), pp. 2714–2720, 2002.

[17] Dias, M. B. and Stenz, A., "A comparative study between centralized, market-based, and behavioral multirobot coordination approaches," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (Las Vegas, USA), pp. 2279–2284, 2003.

[18] Dias, M., *TraderBots: A New Paradigm for Robust and Efficient MultiRobot Coordination in Dynamic Environments*. PhD thesis, Carnegie Mellon University, 2004.

[19] Dias, M., Zinck, M. B., Zlot, R. M., and Stentz, A., "Robust multirobot coordination in dynamic environments," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, (New Orleans, USA), pp. 3435–3442, 2004.

[20] Dias, M., Zlot, R., Karla, N., and Stentz, A., "Market-based multirobot coordination: A survey and analysis," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1257–1270, 2006.

[21] Dudek, G., Jenkin, M., and Milios, E., *Robot Teams: From Diversity to Polymorphism*, ch. A Taxonomy of Multirobot Systems. A. K. Peters Ltd., 2002.

[22] Farinelli, A., Iocchi, L., Nardi, D., and Ziparo, V. A., "Task assignment with dynamic perception and constrained tasks in a multi-robot system," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, (Barcelona, Spain), pp. 1523–1528, 2005.

[23] Farinelli, A., Iocchi, L., Nardi, D., and Ziparo, V. A., "Assignment of dynamically perceived tasks by token passing in multirobot systems," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1271–1288, 2006.

[24] Farinelli, A., Iocchi, L., and Nardi, D., "Multirobot systems: A classification focused on coordination," *IEEE Transactions on Systems, Man and Cybernetics–Part B: Cybernetics*, vol. 34, no. 5, pp. 2015–2028, 2004.

[25] Fua, C.-H. and Ge, S. S., "COBOS: Cooperative backoff adaptive scheme for multirobot task allocation," *IEEE Transactions on Robotics*, vol. 21, no. 6, pp. 1168–1178, 2005.

[26] Fua, C.-H., Ge, S. S., and Lim, K. W., "BOAs: Backoff adaptative scheme for task allocation with fault tolerance and uncertainty management," in *Proceedings of the IEEE International Symposium on Intelligent Control*, (Taipei, Taiwan), pp. 162–167, 2004.

[27] Gerkey, B., Vaughan, R. T., and Howard, A., "The player/stage project: Tools for multi-robot and distributed sensor systems," in *Proceedings of the 11th International Conference on Advanced Robotics (ICAR)*, (Coimbra, Portugal), pp. 317–323, 2003.

[28] Gerkey, B. and Matarić, M. J., "Murdoch: Publish/subscribe task allocation for heterogeneous agents," in *Proceedings of the Fourth International Conference on Autonomous Agents*, (Barcelona, Spain), pp. 203–204, 2000.

[29] Gerkey, B. and Matarić, M. J., "Sold!: Auction methods for multi-robot coordination," *IEEE Transactions on Robotics and Automation, Special Issue on Multi-robot Systems*, vol. 18, no. 5, pp. 758–768, 2002.

[30] Gerkey, B. and Matarić, M. J., "A formal analysis and taxonomy of task allocation in multi-robot systems," *International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, 2004.

[31] Gil, A., Passino, K., Ganapathy, S., and Sparks, A., "Cooperative scheduling of tasks for networked uninhabited autonomous vehicles," in *Proceedings of the IEEE Conference on Decision and Control*, vol. 1, (Maui, USA), pp. 522–527, 2003.

[32] Giord, D., "Weighted voting for replicated data," in *Proceedings of the 7th ACM Symposium on Operating Systems Principles*, (Pacific Grove, USA), pp. 150–162, 1979.

[33] Golfarelli, M., Maio, D., and Rizzi, S., "A task-swap negotiation protocol based on the contract net paradigm," tech. rep., CSITE (Research Centre For Informatics And Telecommunication Systems), University of Bologna, Bologna, Italy, 1997.

[34] Guerrero, J. and Oliver, G., "Multi-robot task allocation method for heterogeneous tasks with priorities," in *Proceedings of the 7th International Symposium on Distributed Autonomous Robotic Systems (DARS)*, (Toulouse, France), 2004.

[35] Howard, A., Seraji, H., and Werger, B., "Global and regional path planners for integrated planning and navigation," *Journal of Robotic Systems*, vol. 22, no. 12, pp. 767–778, 2005.

[36] Howard, A. and Viguria, A., "Controlled reconfiguration of robotic mobile sensor networks using distributed allocation formalisms," in *NASA Science Technology Conference (NSTC)*, (Maryland, USA), 2007.

[37] Hu, X. and Egerstedt, M., "Formation constrained multi-agent control," *IEEE Transactions on Robotics and Automation*, vol. 17, pp. 947–951, December 2001.

[38] Jamil, M., Batta, R., and Malon, D., "The traveling repairperson home base location problem," *Location Science*, vol. 28, no. 2, pp. 150–161, 1994.

[39] Ji, M. and Egerstedt, M., "Role-assignment in multi-agent coordination," *International Journal of Assistive Robotics and Mechatronics*, vol. 7, no. 1, pp. 32–40, 2006.

[40] Karla, N. and Martinoli, A., "A comparative study of market-based and threshold-based task allocation," in *Proceedings of the 8th International Symposium on Distributed Autonomous Robotic Systems (DARS)*, 2006.

[41] Ko, J., Stewart, B., Fox, D., Konolige, K., and Limketkai, B., "A practical decision-theoretic approach to multi-robot mapping and exploration," in *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 3, (Las Vegas, USA), pp. 3232–3238, 2003.

[42] Krieger, M. J. B. and Billeter, J. B., "The call of duty: Self-organized task allocation in a population of up to twelve mobile robots," *Robotics and Autonomous Systems*, vol. 30, no. 1–2, 2000.

[43] Kuhn, H., "The hungarian method for the assignment problem," *Naval Research Logistics Quarterly 2*, pp. 83–97, 1955.

[44] Kurtzberg, J. M., "On approximation methods for the assigment problem," *Journal of the ACM*, vol. 9, no. 4, pp. 419–439, 1962.

[45] Lagoudakis, M., Berhault, M., Koenig, S., Keskinocak, P., and Kleywegt, A., "Simple auctions with performance guarantees for multi-robot task allocation," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 1, (Sendai, Japan), pp. 698–705, 2004.

[46] Lagoudakis, M., Markakis, V., Kempe, D., Keskinocak, P., Koenig, S., Kleywegt, A., Tovey, C., Meyerson, A., and Jain, S., "Auction-based multi-robot routing," in *Proceedings of Robotics: Science and Systems*, (Cambridge, USA), 2005.

[47] Laporte, G. and Nobert, Y., *The Multi-Depot Travelling Salesman Problem*. C.R.T., 1980.

[48] LaValle, S. M. and Kuffner, J. J., "Randomized kinodynamic planning," *International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[49] Lawler, E. L., *The Traveling Salesman Problem*. John Wiley and Sons, 1985.

[50] Lawton, J., Beard, R., and Young, B., "A decentralized approach to formation maneuvers," *IEEE Transactions on Robotics and Automation*, vol. 19, pp. 933–941, December 2003.

[51] Lerman, K., Jones, C., Galstyan, A., and Matarić, M. J., "Analysis of dynamic task allocation in multi-robot systems," *The International Journal of Robotics Research*, vol. 25, no. 3, pp. 225–241, 2006.

[52] Maza, I., Viguria, A., and Ollero, A., "Networked aerial-ground robot system with distributed task allocation for disaster management," in *Proceedings of the IEEE International Workshop on Safety, Security and Rescue Robotics*, (Gaithersburg, USA), 2006.

[53] McLurkin, J. and Yamins, D., "Dynamic task assignment in robot swarms," in *Proceedings of Robotics: Science and Systems*, (Cambridge, USA), 2005.

[54] Moore, B. J. and Passino, K. M., "Distributed task assignment for mobile agents," *IEEE Transactions on Automatic Control*, vol. 52, no. 4, pp. 749–753, 2007.

[55] Mosteo, A. R. and Montano, L., "Comparative experiments on optimization criteria and algorithms for auction based multi-robot task allocation," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, (Rome, Italy), pp. 3345–3350, 2007.

[56] Munkres, J., "Algorithms for the assignment and transportation problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, no. 1, pp. 32–38, 1957.

[57] Nanjanath, M. and Gini, M., "Dynamic task allocation for robots via auctions," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, (Orlando, USA), pp. 2781–2786, 2006.

[58] Nilson, N., *Problem-Solving Methods in Artificial Intelligence*. McGraw-Hill, 1971.

[59] Ollero, A. and Amidi, O., "Predictive path tracking of mobile robots," in *Proceedings of 5th International Conference on Advanced Robotics, Robots in Unstructured Environments (ICAR)*, vol. 2, pp. 1081–1086, 1991.

[60] Parker, L. E., "L-alliance: Task-oriented multirobot learning in behavior-based systems," *Advanced Robotics*, vol. 11, no. 4, pp. 305–322, 1997.

[61] Parker, L. E., "Alliance: An architecture for fault-tolerant multi-robot cooperation," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 2, pp. 220–240, 1998.

[62] Passino, K. M. and Yurkovich, S., *Fuzzy Control*. Addison-Wesley, 1998.

[63] Report, N., "National workshop on future sensing systems - living, nonliving, and energy systems," tech. rep., Lake Tahoe, NV, 2002.

[64] Rosenblatt, J. K., "Damn: a distributed architecture for mobile navigation," *Journal of Experimentation and Theoretical Artificial Intelligence*, vol. 9, no. 2, pp. 339–360, 1997.

[65] Sandholm, T., "An implementation of the contract net protocol based on marginal cost calculations," in *Proceedings of the 12th International Workshop on Distributed Artificial Intelligence*, (Hidden Valley, USA), pp. 295–308, 1993.

[66] Sariel, S. and Balch, T., "A distributed multi-robot cooperation framework for real time task achievement," in *Proceedings of the 8th International Symposium on Distributed Autonomous Robotic Systems (DARS)*, 2006.

[67] Sariel, S. and Balch, T., "Efficient bids on task allocation for multi-robot exploration," in *Proceedings of the 19th International FLAIRS Conference*, (Melbourne Beach, USA), 2006.

[68] Sariel, S., Balch, T., and Erdogan, N., "Robust multi-robot cooperation through dynamic task allocation and precaution routines," in *The 3rd International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, (Minneapolis, USA), 2006.

[69] Sariel, S., Balch, T., and J., S., "Empirical evaluation of auction-based coordination of auvs in a realistic simulated mine countermeasure task," in *Proceedings of the 8th International Symposium on Distributed Autonomous Robotic Systems (DARS)*, (Minneapolis, USA), 2006.

[70] Sariel, S., Balch, T., and Erdogan, N., "Naval mine countermeasure missions," *IEEE Robotics and Automation Magazine*, vol. 15, no. 1, pp. 45–52, 2008.

[71] Scerri, P., Farinelli, A., Okamoto, S., and Tambe, M., "Token approach for role allocation in extreme teams," in *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, (The Netherlands), pp. 727–734, 2005.

[72] Smith, G., "The contract net protocol: High-level communication and control in a distributed problem solver," *IEEE Transactions on Computers*, vol. C-29, no. 12, pp. 1104–1113, 1980.

[73] Smith, S. L. and Bullo, F., "Target assignment for robotic networks: asymptotic performance under limited communication," in *Proceedings of the American Control Conference*, (New York, USA), pp. 3585–3590, 2007.

[74] Thrun, S., Burgard, W., and Fox, D., *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[75] Toroslu, I. H. and Üçoluk, G., "Incremental assignment problem," *Information Sciences*, vol. 177, no. 6, pp. 1523–1529, 2007.

[76] Tovey, C., Lagoudakis, M. G., Jain, S., and Koenig, S., *Multi-Robot Systems. From Swarms to Intelligent Automata Volume III, Proceedings from the 2005 International Workshop on Multi-Robot Systems*, ch. The Generation of Bidding Rules for Auction-Based Robot Coordination, pp. 3–14. Springer Netherlands, 2005.

[77] Viguria, A. and Howard, A., "Upper bound analysis of a cost market-based algorithm applied to the initial formation problem," in *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, (San Diego, USA), pp. 2326–2331, 2007.

[78] Viguria, A., Maza, I., and Ollero, A., "SET: An algorithm for distributed multirobot task allocation with dynamic negotiation based on task subsets," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, (Rome, Italy), pp. 3339–3344, 2007.

[79] Werger, B. B. and Matarić, M. J., "Broadcast of local eligibility for multi-target observation," in *Distributed Autonomous Robotic Systems 4*, pp. 347–356, Springer-Verlag, 2000.

[80] Williams, S. and Howard, A., "A single camera terrain slope estimation technique for natural arctic environments," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, (Pasadena, USA), 2008.

[81] Xu, Y., Scerri, P., Sycara, K., and Lewis, M., "Comparing market and token-based coordination," in *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, (Hakodate, Japan), pp. 1113–1115, 2006.

[82] Xu, Y., Scerri, P., Yu, B., Okamoto, S., Lewis, M., and Sycara, K., "An integrated token-based algorithm for scalable coordination," in *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, (The Netherlands), pp. 407–414, 2005.

[83] Yun, S. and Rus, D., "Optimal distributed planning of multi-robot placement on a 3d truss," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (San Diego, USA), pp. 1365–1370, 2007.

[84] Zavlanos, M. M. and Pappas, G. J., "Dynamic assignment in distributed motion planning with limited information," in *Proceedings of the American Control Conference*, (New York, USA), pp. 1173–1178, 2007.

[85] Zlot, R. and Stentz, A., "Market-based multirobot coordination for complex tasks," *International Journal of Robotics Research Special Issue on the 4th International Conference on Field and Service Robotics*, vol. 25, no. 1, pp. 73–101, 2006.